AN LLL REDUCTION AIDED SPHERE DECODING ALGORITHM

INTEGER LEAST SQUARES PROBLEMS

APPLICATION IN MIMO SYSTEMS: AN LLL REDUCTION AIDED SPHERE DECODING ALGORITHM

By

JIN CAI GUO, H.BSc

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Science

McMaster University

©Copyright by Jin Cai Guo, April 2008

MASTER OF SCIENCE (2008) McMaster University

COMPUTING AND SOFTWARE Hamilton, Ontario

TITLE: Integer Least Squares Problem Application in MIMO systems: An LLL Reduction Aided Sphere Decoding Algorithm

AUTHOR: Jin Cai Guo, H.BSc. with Distinction (University of Toronto)

SUPERVISOR: Dr. Sanzheng Qiao

NUMBER OF PAGERS: xi, 136

Abstract

Solving the integer least squares problem $\min \|H\mathbf{s} - \mathbf{x}\|_2$, where the unknown vector $\mathbf{s}$ is comprised of integers, the coefficient matrix H and given vector $\mathbf{x}$ are comprised of real numbers arises in many applications and is equivalent to find the closest lattice point to a given one known as **NP**-hard. In multiple antenna systems, the received signal represented by vector $\mathbf{x}$ is not arbitrary, but an lattice point perturbed by an additive noise vector whose statistical properties are known. It has been shown the Sphere Decoding, in which the lattice points inside a hyper-sphere are generated and the closest lattice point to the received signal is determined, together with Maximum Likelihood (ML) method often yields a near-optimal performance on average (cubic) while the worst case complexity is still exponential. By using lattice basis reduction as pre-processing step in the sub-optimum decoding algorithms, we can show that the lattice reduction aided sphere decoding (LRSD) achieves a better performance than the maximum likelihood sphere decoding (MLSD) in terms of symbol error rate (SER) and average algorithm running time. In the FIR (Finite Impulse Response) MIMO channel, the channel matrix is Toeplitz and thus gives us the leverage to use the fact that all its column vectors all linearly independent and the matrix itself is often well-conditioned.

In this thesis, we will develop a lattice reduction added sphere decoding algorithm along with an improved LLL algorithm, and provide the simulations to show that this new algorithm achieves a better performance than the max-

imum likelihood sphere decoding.

In chapter 1, we define our system model and establish the foundations for understanding of mathematical model – namly the integer least squares problem, and thus the choice of the simulation data. In chapter 2, we explain the integer least squares problems and exploit serveral ways for solving the problems, then we introduce the sphere decoding and maximum likelihood at the end. In chapter 3, we explore the famous LLL reduction algorithm named after Lenstra, Lenstra and Lovasz in details and show an example how to break Merkle-Hellman code using the LLL reduction algorithm. Finally, in chapter 4 we give the LLL reduction aided sphere decoding algorithm and the experiment setup as well as the simulation results against the MLSD and conclusions, further research directions.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisor, Dr. Sanzheng Qiao, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way. I attribute the level of my Masters degree to his remarkable guidance and encouragement and without him this thesis, too, would not have been completed or written. One simply could not wish for a better or friendlier supervisor. His enthusiasm for providing high-quality work has always been my deepest impression.

I would also like to thank my defense committee members, Dr. Antoine Deza and Dr. Kamran Sartipi, for their valuable comments to my work. I indebt to Jing Liu, a Phd student at the ECE department specialize in the wireless communication, coding and decoding theory, without her, it would not possible to run a justified simulations and so are the obtained data.

Last but not least, I must thank my wife for her never-ending love and support. She has been taking care of our financial needs all along so that I could concentrate on my research work. I hope that this work can be the least reward to her efforts.

Hamilton, Ontario, Canada — Jin Cai Guo

March, 2008

Contents

List of Figures

Chapter 1

Introduction to System and Channel Model

Over the past few year, it has been shown that using multiple antennas can significantly increase the capacity and robustness of communication systems in fading environments. As a result, much work has been done toward designing coding and decoding schemes to realize these gains promised by theoretical studies [5, 15, 34]. Recently, the use of the sphere decoding for lattice codes was proposed in [47] and has been shown in [20] that the average complexity of the sphere decoding used for maximum-likelihood detection in dispersive multiple antenna systems is polynomial (often cubic) with respect to a wide range of signal-to-noise ratios (SNRs). In this thesis, we will develop a new algorithm – a lattice reduction aided sphere decoding algorithm. But first, let us refresh our memories about MIMO multiple antenna systems and then

define the channel and system model used in this thesis.

1.1 Preliminaries on MIMO multiple antenna systems

Multiple input multiple output (MIMO) channels, or Vector channels, represent a very general description for a wide range of applications. They incorporate SISO (Single Input Single Output), MISO (Multiple Input Single Output) and SIMO (Single Input Multiple Output) channels as special cases. Often, MIMO channels are only associated with multiple antenna systems.

There are many reasons to use multiple antenna systems, two principle ones are: [29]

1. Multiple antenna systems improve the link reliability. For example, if multiple access or cochannel interference in cellular networks disturbs the transmission, interferes that are separable in space can be suppressed with multiple antennas, resulting in an improved signal to interference plus noise ratio (SINR).

2. Multiple antenna systems are capable of providing high data transmissions over wireless channels.

Research in wireless communications mainly studies transmissions over time-invariant channels, such as frequency-selective channel which arises at

high data rates and causes subsequently transmitted data symbols to interfere with each other (also known as intersymbol interference (ISI)). Communication systems transmitting over frequency-selective channels generally employ an equalizer to recover the transmitted sequence corrupted by intersymbol interference. Most practical systems use a training sequence to learn the channel impulse response and thereby design the equalizer. The channel is often assumed to be discrete-time finite-impulse-response (FIR), subject to block fading, i.e., the channel impulse response is constant over an interval after which is changed to an independent value.

Let us assume a discrete-time block-fading frequency-selective **FIR** channel model, $s(n)$ denotes the present value of the input, and $s(n-1), s(n-2), \ldots, s(n-L)$ denote the L past values of the input. Let $y(n)$ denote the present value of the channel output. we can then describe the input-output relation of the channel by the convolution sum [37]

$$y(n) = \sum_{k=0}^{L} h(k)s(n-k), \tag{1.1}$$

thus the z-transformation of the channel impulse response is given by

$$H(z) = h_1 + h_2 z^{-1} + \cdots + h_L z^{-L+1}, \tag{1.2}$$

where L is also called the channel order and the channel coefficient $\{h_i\}_{i=1}^{L}$ are constant for some discrete interval of T channel uses, after which they may change to independent values held constant for another interval of length T,

and so on.

In the mathematical abstraction, the domain of a discrete-time signal is the set of integers (or some interval). What these integers represent depends on the nature of the signal. The transmission of the signals can be divided into two phases: the training phase and the data transmission phase.

1.1.1 Training Phase

During the training phase we transmit the T_τ training symbol $\theta_1, \theta_2, \ldots, \theta_{T_\tau}$, since we are interested in estimating the L channel coefficients $h_1, h_2, \ldots, h_L$, to obtain meaningful estimates we require $T_\tau \geq L.$, which provides the receiver with at least as many equations as there are unknowns. To allow the transmission of data, we clearly also require $T_\tau < T$. We can write:

$$y_\tau = \delta_\tau \theta_\tau h + v_\tau \tag{1.3}$$

$v_\tau \in \mathbb{R}^{T_\tau \times 1}$ is a vector of independent zero-mean unit variance additive Gaussian noise, and δ_τ^2 is the expected transmit energy during the training phase, the training symbol vector $\theta_\tau = [\theta_1, \theta_2, \ldots, \theta_{T_\tau}]^T$ satisfies the power constraint $tr(\theta_\tau)^2 = T_\tau$. Furthermore,

$$\theta_\tau = \begin{bmatrix} \theta_1 & 0 & 0 & \dots & 0 \\ \theta_2 & \theta_1 & 0 & \dots & 0 \\ \theta_3 & \theta_2 & \theta_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \theta_L & \theta_{L-1} & \theta_{L-2} & \dots & \theta_1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \theta_{T_\tau} & \theta_{T_\tau-1} & \theta_{T_\tau-2} & \dots & \theta_{T_\tau-L+1} \end{bmatrix}_{T_\tau \times L}.$$

1.1.2 Data Transmission Phase

The data transmission phase consists of $T_d > 0$ channel uses. If we collect the transmitted symbols into the T_d-dimensional column vector $s_d = [s_1, s_2, \dots, s_{T_d}]^T$, and the received signals into the $(T_d + L - 1)$-dimensional column vector $y_d = [y_1, y_2, \dots, y_{T_d+L-1}]^T$, then we can write:

$$y_d = H_d s_d + v_d \tag{1.4}$$

$H_d \in \mathbb{R}^{(T_d+L-1)\times T_d}$, $v_d \in \mathbb{R}^{T_d+L-1}$ is a vector of independent zero-mean unit variance additive Gaussian noise. Furthermore,

$$H = \begin{bmatrix} h_1 & 0 & \dots & 0 \\ h_2 & h_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_L & & & h_1 \\ 0 & h_L & & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & h_L \end{bmatrix}. \tag{1.5}$$

Notice that, compared to the training phase, the channel coefficients are known while the transmitted symbols are unknown in the data transmission phase.

Having introduced the MIMO multiple antenna system and the frequency selective channel, let us describe in the following section the channel and system model that is going to be used in this thesis.

1.2 System and Channel Description

Consider a multiple antenna system shown in Figure (1.1) with M transmit antennas and N receive antennas. At each time instant, say kth instant, M signals $(s_k^{(1)}, s_k^{(2)}, \dots, s_k^{(M)})$, satisfying an average power constraint, are transmitted using M antennas and reaches all N receive antennas. In this thesis, we model the channel as frequency selective, Rayleigh and block fading, with channel knowledge at the receiver and additive white Gaussian noise(AWGN),

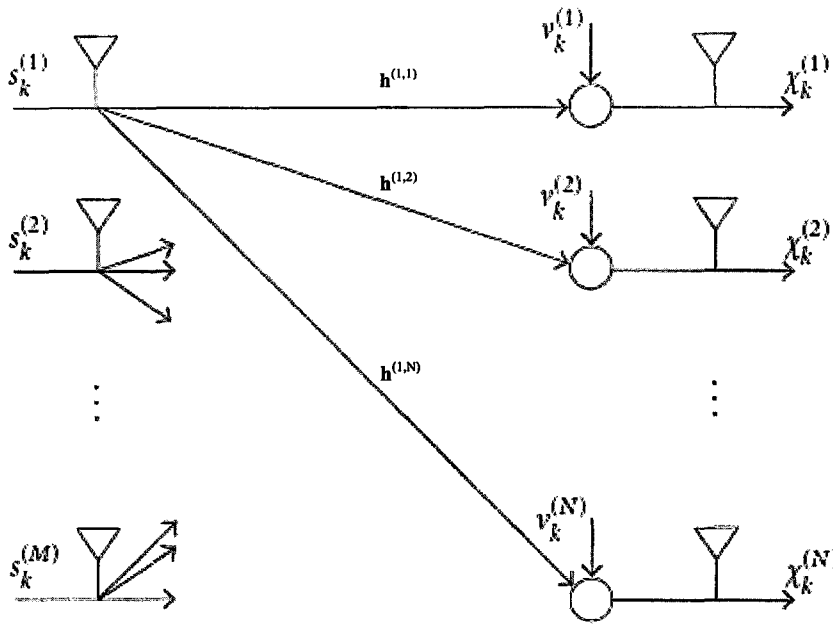

Figure 1.1: FIR MIMO channel model

and we explain these terms as followings [37, 39]:

- **Frequency Selective Fading:** due to time dispersion of the transmitted symbols within the channel, if the channel possesses a constant gain and linear phase response over a bandwidth that is smaller than the bandwidth of transmitted signal, then the channel creates such fading on the received signal. When this occurs, the received signal includes multiple versions of the transmitted waveform which are attenuated(faded) and delayed in time, and hence the received signal is distorted. In any radio transmission, the channel spectral response is not flat due to reflections and scattering which are the reasons cause the fading. This assumption is practical as multipath propagation is the typical charac-

teristics of wireless communication channels.

- **Rayleigh Fading:** the channel coefficients h_{ij} are independent and identically distributed with zero mean, unit variance, circularly symmetric, Gaussian density ($\mathcal{CN}(0,1)$). Notice, however, the channel coefficients are correlated in most environments, the correlation is less when the antennas are well separated and there are a large number of scatters in the environment. Nevertheless, Rayleigh fading model is more often used in theoretical analysis as it offers a nice statistical property for the channel coefficient(s).

- **Block Fading:** the channel stays fixed for a certain period, called the coherence time of the channel, and then changes to something independent for the next block. Notice that in reality, channel coefficients change gradually from one time to another. However, block fading is often used in theoretical analysis for its simplicity.

- **Channel Knowledge at Receiver:** perfect channel knowledge is available at the receiver(s) but not the transmitter(s), or coherent detection. In practices, the receiver(s) can not know the channel perfectly. However, if the channel varies slowly, we can assume the receiver(s) has sufficient time to get a good estimation of the channel.

- **AWGN at Receiver:** At each receiver, signals received from all transmit antennas are added together, along with iid additive white Gaussian

noise with zero mean and variance $\mathcal{CN}(0, \delta^2)$.

Finally, we assume that the system feedback is unavailable and the receiver learns the channel based on the training information, and we also restrict our attention to the case where the code duration is shorter than the coherence time of the channel, so that each codeword experiences only one channel realization. With the above channel model, let the column vector

$$h^{(i,j)} = [h_1^{(i,j)}\ h_2^{(i,j)}\ \ldots h_L^{(i,j)}\ ]^T$$

denotes the single-input single-output (SISO) channel impulse response from the j^{th} transmit antenna to the i^{th} receive antenna, where T is the transpose operation and L is the channel order as defined before. Then from the (1.1), the received signal at the i^{th} antenna can be expressed as

$$x_k^{(i)} = \sum_{j=1}^{M} \sum_{l=1}^{L} h_l^{(i,j)} s_{k-l}^{(j)} + v_k^{(i)}.$$

The above equation can be written as

$$\mathbf{x}_k = \sum_{l=1}^{L} H_l \mathbf{s}_{k-l} + \mathbf{v}_k, \tag{1.6}$$

- $\mathbf{v}_k$ is the additive noise vector defined as $\mathbf{v}_k = [v_k^{(1)}, v_k^{(2)}, \ldots, v_k^{(N)}]^T$,
- $\mathbf{s}_{k-l} = [s_k^{(1)}, s_k^{(2)}, \ldots, s_k^{(M)}]^T$ is the transmitted signal vector, whose entries typically come from a QAM ro PAM constellation, and

- $H_l \in C^{N \times M}$ is the l^{th} coefficient matrix in the MIMO channel impulse response

$$H_l = \begin{bmatrix} h_l^{(1,1)} & h_l^{(1,2)} & \dots & h_l^{(1,M)} \\ h_l^{(2,1)} & h_l^{(2,2)} & \dots & h_l^{(2,M)} \\ \vdots & \vdots & \ddots & \vdots \\ h_l^{(N,1)} & h_l^{(N,2)} & \dots & h_l^{(N,M)} \end{bmatrix}.$$

Define

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{T+L-1}]^T,$$
$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_{T+L-1}]^T,$$
$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{x}_T]^T,$$

we can write the input-output relation for the FIR MIMO channel in the matrix form as [46]

$$\mathbf{X} = H\mathbf{S} + \mathbf{V}, \tag{1.7}$$

where $H \in C^{N(T_d+L-1)\times MT_d}$ is constructed as

$$H = \begin{bmatrix} H_1 & 0 & \dots & 0 \\ H_2 & H_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H_L & & & H_1 \\ 0 & H_L & & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & H_L \end{bmatrix}. \tag{1.8}$$

Notice all quantities in (1.7) are complex, so one may find it is useful to rewrite it in terms of real quantities. So, let us define

$$\mathbf{x} = [\Re(\mathbf{X}) \quad \Im(\mathbf{X})]^T,$$
$$\mathbf{v} = [\Re(\mathbf{V}) \quad \Im(\mathbf{V})]^T,$$

and

$$H = \begin{bmatrix} \Re(H) & -\Im(H) \\ \Im(H) & \Re(H) \end{bmatrix}.$$

Then (1.7) can be rewritten as

$$\mathbf{x} = H\mathbf{s} + \mathbf{v}, \tag{1.9}$$

where $\mathbf{x} \in \mathbb{R}^{2NT_d \times 1}$, $\mathbf{v} \in \mathbb{R}^{2NT_d \times 1}$, and $H \in \mathbb{R}^{2N(T_d+L-1) \times 2MT_d}$ is still a Toeplitz matrix.

For simplicity, we assume all the quantities in (1.7) are real, which is not unrealistic if considering the PAM constellation, and focus on the SISO case, which can serve as building blocks for complex MIMO systems. Additionally, we also assume that the channel coefficients obtained through the training sequence are exact, i.e., there is not errors in the estimation of these variables.

With the notations and restrictions introduced above, due to the existence of Gaussian additive noise in the channel, we can express the transmitted signal recovery problem as the optimization problem

$$\min_{s \in \mathbb{Z}^{T_d}} \|\mathbf{x} - H\mathbf{s}\|^2,$$

where the minimization is over all the integer points in $\mathbb{Z}^{T_d}$. This is indeed a integer least square problem (ILS) to find the closest lattice point to a given T dimensional vector $\mathbf{x}$ in the skewed lattice $H\mathbf{s}$, which is known to be NP hard. As the channel matrix is in fact obtained through training sequences, perturbation on the channel matrix H has to be considered in practice. In the following chapter, we will introduce the Integer Least Squares (ILS) problem and the maximum likelihood sphere decoding algorithm.

Chapter 2

Integer Least Squares Problem and Maximum Likelihood Sphere Decoding

The integer least squares (ILS) problem is an extension of the least squares (LS) problem in the integer domain for solving the problem of finding a vector $\mathbf{s} \in \mathbb{Z}^m$ such that $\min_{\mathbf{s}\in\mathbb{Z}^m} \|H\mathbf{s} - \mathbf{x}\|_2$, where the data matrix $H \in \mathbb{R}^{n\times m}$ and the observation vector $\mathbf{x} \in \mathbb{R}^n$ are given and $n \geq m$. In this chapter, we are going to discuss the integer least squares problems and general methods used to solve them, their applications in wireless communication, and finally the maximum likelihood sphere decoding algorithm.

2.1 Introduction to the Least Squares Problems

Consider the problem of finding the least-squares solution to the problem:

$$\min \|H\mathbf{s} - \mathbf{x}\|_2 \qquad (*)$$

where H is an $n \times m$ matrix($n \geq m$) and $\mathbf{x}$ is an n-vector, both are with real entries, is more precisely to minimize the Euclidean norm of the residual $H\mathbf{s} - \mathbf{x}$. That is,

$$\begin{aligned} \|H\mathbf{s} - \mathbf{x}\|^2 &= (H\mathbf{s} - \mathbf{x})^T (H\mathbf{s} - \mathbf{x}) \\ &= (H\mathbf{s})^T (H\mathbf{s}) - x^T H\mathbf{s} - (H\mathbf{s})^T \mathbf{x} + \mathbf{x}^T\mathbf{x}. \end{aligned}$$

Notice the two middle terms are equal, and the minimum is found at the zero of the derivative with respect to $\mathbf{s}$:

$$2H^T H\mathbf{s} - 2H^T\mathbf{x} = 0,$$

we can solve it to get the vector $\mathbf{s}$ as the solution of the normal equations $H^T H\mathbf{s} = H^T\mathbf{x}$. The above introduced method is called normal equations method and is the most widely used method for solving the full rank LS problem, but the accuracy of the computed normal equations solution depends

on the square of the codition number of H. If we define the minimum residual

$$\epsilon_{LS} = \|(I - H(H^T H)^{-1} H^T)\mathbf{x}\|_2,$$

then we can conclude that if ϵ_{LS} is small and the condition number of H is large, then the method of normal equations does not solve a nearby problem and will usually render an LS solution that is less accurate than a stable QR approach. On the other hand, the normal equation and QR methods produce comparably inaccurate results when applied to large residual, ill-conditioned problems [19].

At this point we have to also mention the total least squares (TLS) problems. The total least squares problem minimizes the Frobenius norm of the correction matrix $[\Delta H;\ \Delta\mathbf{x}]$, that is

$$\min\|[\Delta H \quad \Delta\mathbf{x}]\|_F \quad \text{subject to} \quad (H + \Delta H)\mathbf{s} = \mathbf{x} + \Delta\mathbf{x}.$$

where H is an $n \times m$ matrix($n \geq m$) and $\mathbf{x}$ is an n-vector, both are with real entries, $[\Delta H;\ \Delta\mathbf{x}]$ is called the corresponding TLS correction [23].

One important application of TLS problems is parameter estimation in errors-in-variables models. Let us define an overdetermined linear equations system with n measurements in H, $\mathbf{x}$ and m unknown parameters $\mathbf{s}$ by

$$H_0\mathbf{s} = \mathbf{x}_0$$

$$H = H_0 + \Delta H$$

$$\mathbf{x} = \mathbf{x}_0 + \Delta \mathbf{x},$$

where ΔH and $\Delta \mathbf{x}$ are the measurement errors. A reasonable assumption is that H has full rank and all rows of $[\Delta H;\ \Delta \mathbf{x}]$ are independently and identically distributed with zero mean and covariance matrix $\delta^2 I$, as this coincides with our system model defined in previous chapter, i.e., H is Toeplitz and the channel noise is iid with zero means and covariance δ^2. Then it can be proved [17] that the TLS solution $\hat{\mathbf{s}}$ of $H\mathbf{s} \approx \mathbf{x}$ estimates the *true* parameter values $\mathbf{s}_0$ given by $(H^T H_0)^{-1}\mathbf{x}_0$, where -1 is the matrix inverse operation, as n tends to infinity. This property of TLS estimation does not depend on any assumed distributional form of the errors. Had we not assumed exact channel knowledge, we would have to explore the TLS model as well.

2.2 Least Squares Solutions

The least squares problems have several explicit solutions including normal equations, QR decomposition and SVD (singular value decomposition). The first method is the fastest but least accurate, it is adequate when the condition number is small. The second method is the standard one but slower than the first one. The third method is of most use on ill-conditioned problem, i.e., when H is not of full rank, it is more computation intensive than the QR. Of all the methods used for solving the LS problems, QR factorization and SVD

(singular value decomposition) are the most common and widely used. In the following, we will briefly introduce these two methods.

2.2.1 QR decomposition

the QR decomposition (also called the QR factorization) of a matrix is a decomposition of the matrix into an orthogonal and a triangular matrix Q and R respectively. The QR decomposition is often used to solve the linear least squares problem, and is defined in the following theorem:

Theorem 2.2.1 (QR Decomposition (QR)) *Let H be a real n-by-m matrix. Suppose H has full column rank, then there exist a unique $n \times m$ orthogonal matrix Q such that $Q^TQ = I_m$, and a unique $m \times m$ upper triangular matrix R with positive diagonals $r_{ii} > 0$ such that $H = QR$.*

***Proof*:** Omitted, please see [11] on pp.107.

There are several methods for computing the QR decomposition, such as Gram-Schmidt, Householder transfromation or Givens rotations. Each has a number of advantages and disadvantages. Here we will only introduce the Householder transformation and Gram-Schmidt methods as they will be used later in this thesis.

2.2.1.1 Householder Transformation

A Householder transformation is an $n \times n$ matrix of the form [22]

$$P = I - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}$$

where $\mathbf{v}$ is a nonzero vector. It is easy to see that $P = P^T = P^{-1}$ from the definition, so that P is both orthogonal and symmetric.

Given a vector $\mathbf{u}$, we can find a Householder transformation P to zero out all but the first entry of $\mathbf{u}$, i.e.,

$$P\mathbf{u} = c \cdot \mathbf{e_1}.$$

Using the formula for P, we have

$$c \cdot \mathbf{e_1} = P\mathbf{u} = \left(I - 2\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\right)\mathbf{u} = \mathbf{u} - 2\mathbf{v}\frac{\mathbf{v}^T\mathbf{u}}{\mathbf{v}^T\mathbf{v}},$$

hence we can take

$$\mathbf{v} = \mathbf{u} - c\mathbf{e_1}.$$

We also must have $c = \pm\|\mathbf{u}\|_2$ to preserve the norm, and the sign should be chosen to avoid cancellation, i.e., $c = -\text{sign}(u_1)\|\mathbf{u}\|_2$. Without going into too much details, we hereby give the Householder QR factorization (for more details, please see [19] in chapter 5):

Algorithm: Householder Vector

Input: Given $\mathbf{u} \in \mathbb{R}^n$.

Output: $\mathbf{v} \in \mathbb{R}^n$ with $v_1 = 1$, and $c \in \mathbb{R}$ such that $P = I_n - c\mathbf{v}\mathbf{v}^T$ is orthogonal and $P\mathbf{u} = \|\mathbf{u}\|_2 e_1$.

function $[\mathbf{v}, c] = house(\mathbf{u})$

$n = length(\mathbf{u})$

$\sigma = \mathbf{u}(2:n)^T \mathbf{u}(2:n)$

$\mathbf{v} = [1, \mathbf{u}(2:n)]^T$

if $\sigma = 0$

$c = 0$

else

$\mu = \sqrt{\mathbf{u}(1)^2 + \sigma}$

if $\mathbf{u}(1) \leq 0$

$\mathbf{v}(1) = \mathbf{u}(1) - \mu$

else

$\mathbf{v}(1) = -\sigma/(\mathbf{u}(1) + \mu)$

end

$c = 2\mathbf{v}(1)^2/(\sigma + \mathbf{v}(1)^2)$

$\mathbf{v} = \mathbf{v}/\mathbf{v}(1)$

end

Notice that $c = 2/\mathbf{v}^T\mathbf{v}$ in the above algorithm. Now, we can subsequently choose a Householder matrix P_i to zero out hte subdiagonal entries

in each column i without disturbing the zeros already introduced in previous columns.

In general, the matrix, H is overwritten by Q, R instead of outputting matrices Q and R. Since for a $n \times m$ matrix H, the ith Householder vector $\mathbf{v}$ can be written as

$$\mathbf{v}^{(i)} = [\underbrace{0, \ldots, 0}_{j-1}, 1, v_{j+1}^{(i)}, \ldots, v_n^{(i)}]^T, \tag{2.1}$$

so the Householder QR factorization of H can be stored as a upper triangular matrix and a collection of the Householder vectors in the lower triangluar matrix. If the orthogonal matrix Q is needed, we can easily construct it by letting $P_i = I_{n\times n} - c\mathbf{v}^{(i)}(\mathbf{v}^{(i)})^T$ with $\mathbf{v}^{(i)}$ defined in (2.1), and the orthogonal matrix $Q = P_1^T \ldots P_{n-1}^T P_n^T$ as the process above described is of the form

$$P_n P_{n-1} \ldots P_1 H = R.$$

To this end, we complete the QR factorization by explicitly giving an algorithm to construct the upper triangular matrix R and Householder vectors. We do this in the following algorithm.

Algorithm: QR decomposition using Householder transformations

for $i = 1$ to $\min(m, n-1)$

$[\mathbf{v}, c] = house(H(i:n, i))$

$H(i:n, i:m) = (I_{n-i+1} - c\mathbf{v}\mathbf{v}^{\mathbf{T}})H(i:n, i:m)$

$$H(i+1:n,i) = \mathbf{v}(2:n-i+1)$$

end

There are variant versions of QR factorizations, such as Householder QR with column pivoting, but we will not further explore them here.

2.2.1.2 Gram-Schmidt Orthogonalization

The Gram-Schmidt orthogonalization is one of alternative methods to compute the QR decomposition with two variations: *Classical Gram-Schmidt* (CGS) and *Modified Gram-Schmidt* (MGS).

Write the matrix H as a sequence of vectors $\{\mathbf{h}_1|\mathbf{h}_2|,\ldots,|\mathbf{h}_n\}$, then

$$\mathbf{u}_1 = \mathbf{h}_1, \quad \mathbf{e}_1 = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}$$

$$\mathbf{u}_2 = \mathbf{h}_2 - Proj_{\mathbf{u}_1}\mathbf{h}_2, \quad \mathbf{e}_2 = \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}$$

...

$$\mathbf{u}_n = \mathbf{h}_n - \Sigma_{i=1}^{n-1} Proj_{\mathbf{u}_i}\mathbf{h}_n, \quad \mathbf{e}_n = \frac{\mathbf{u}_n}{\|\mathbf{u}_n\|}$$

where $Proj_{\mathbf{u}}\mathbf{h} = \frac{<\mathbf{h},\mathbf{u}>}{<\mathbf{u},\mathbf{u}>}\mathbf{u}$ is called projection of $\mathbf{h}$ on $\mathbf{u}$.

This is equivalent to say

$$\begin{aligned}\mathbf{u}_j &= \mathbf{h}_j - (\mathbf{q}_1^T\mathbf{h}_1)\mathbf{q}_1 - \cdots - (\mathbf{q}_{j-1}^T\mathbf{h}_{j-1})\mathbf{q}_{j-1} \\ &= \mathbf{h}_j - \Sigma_{i=1}^{j-1}(\mathbf{q}_i^T\mathbf{h}_i)\mathbf{q}_i,\end{aligned}$$

where $\mathbf{q}_i = \mathbf{u}_i / \|\mathbf{u}_i\|$ is a normalized vector. Clearly, all the $\mathbf{q}_i$s together with the $\mathbf{q}_n = \mathbf{u}_i / \|\mathbf{u}_i\|$ are the columns of the orthogonal matrix Q. If we write $R = Q^T H$, then

$$r_{ij} = \mathbf{q}_i^T \mathbf{h}_j, \quad \text{if}(i < j)$$

and

$$r_{jj} = \|\mathbf{h}_j - \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i\|_2.$$

Thus, the Classical Gram-Schmidt algorithm can be written as:

for $j = 1$ **to** n

  $\mathbf{u}_j = \mathbf{h}_j$

  for $i = 1$ **to** $j - 1$

    $r_{ij} = \mathbf{q}_i^T \mathbf{h}_j$

    $\mathbf{u}_j = \mathbf{u}_j - r_{ij}\mathbf{q}_i$

  end

  $r_{jj} = \|\mathbf{u}_j\|_2$

  $\mathbf{q}_j = \mathbf{u}_j / r_{jj}$

end

Unfortunately, the CGS method is less satisfactory when implemented in finite precision arithmetic, as orthogonality among the computed $\mathbf{q}_i$ tends to be lost due to rounding error. A rearrangement of the calculation, known as *modified Gram-Schmidt* (MGS), yields a better results [45]. The idea is simple:

if we know $\mathbf{q}_i$, then we can apply q_i to update $\mathbf{h}_i$s. So the MGS algorithm can be written as:

for $i = 1$ **to** n

  $\mathbf{u}_i = \mathbf{h}_i$

end

for $i = 1$ **to** n

  $r_{ii} = \|\mathbf{u}_i\|$

  $\mathbf{q}_i = \mathbf{u}_i / r_{ii}$

  for $j = i + 1$ **to** n

    $r_{ij} = \mathbf{q}_i^T \mathbf{h}_j$

    $\mathbf{u}_j = \mathbf{u}_j - r_{ij}\mathbf{q}_i$

  end

end

The MGS method is equivalent mathematically, but superior numerically, to CGS. However, it is slightly more expensive than Householder QR as it always manipulates m-vectors whereas the latter procedure deals with ever shorter vectors. As with square linear system, a diagonal linear squares problem is even easier to solve than a triangular one. The singular value decomposition (SVD) is such a method that goes beyond the triangular QR factorization to achieve a diagonal factorization of a rectangular matrix using orthogonal transformations. In the following, we will briefly introduce the SVD.

2.2.2 Singular Value Decomposition

A general way to find a least squares solution to an overdetermined system is to use a singular value decomposition to form a matrix that is known as the pseudo-inverse of a matrix, this technique works even if the matrix is rank deficient. The numerical analysis for the algorithm, which is based on the SVD used in [18], reveals that the SVD is numerically robust but computationally expensive. The SVD plays an important role in a number of matrix approximation problems. Among other things, the SVD enables us to intelligently handle the matrix rank problem and reveals a great deal about the structure of a matrix.

Theorem 2.2.2 (Singular Value Decomposition (SVD)) *if H is a real n-by-m matrix, then there exist orthogonal matrices*

$$U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n \times n}$$

and

$$V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m] \in \mathbb{R}^{m \times m}$$

such that

$$U^T H V = diag(\sigma_1, \sigma_2, \ldots, \sigma_p) \in \mathbb{R}^{n \times m} \quad p = min\{n, m\}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

***Proof*:** Omitted, please see [19] on pp.70.

Furthermore, if $H = U\Sigma V^T \in \mathbb{R}^{n\times m}$ is the SVD of H and $n \geq m$, then

$$H = U_1\Sigma_1 V^T,$$

where

$$U_1 = U(:, 1:m) = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m] \in \mathbb{R}^{n\times m}$$

and

$$\Sigma_1 = \Sigma(1:m, 1:m) = diag(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{R}^{m\times m}.$$

We refer to this much-used, trimmed down version of the SVD as the *thin* SVD.

Let us now define the minimum norm LS soution of $H\mathbf{s} \approx x$ using the SVD in the following.

Theorem 2.2.3 (Minimum Norm LS Soution of $H\mathbf{s} \approx x$) *Let the SVD of $H \in \mathbb{R}^{n\times m}$ be given by $H = U\Sigma V^T$, where $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n] \in \mathbb{R}^{n\times n}$ is an unitary matrix, $\Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{R}^{n\times m}$ is a matrix with nonnegative numbers on the diagonal such as $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$ and zeros off the diagonal, and V^T is the transpose of the unitary matrix $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m] \in \mathbb{R}^{m\times m}$, i.e., $H = \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$, and assume that $rank(H) = r$. if $\mathbf{x} \in \mathbb{R}^n$, then*

$$\hat{\mathbf{x}} = \sum_{i=1}^{r} \sigma_i^{-1} \mathbf{v}_i \mathbf{u}_i^T b$$

minimizes $\|H\mathbf{s}-\mathbf{x}\|_2$ *and has the smallest 2-norm of all minimizers. Moreover,*

$$\rho^2 = \|H\mathbf{s} - \mathbf{x}\|_2^2 = \sum_{i=r+1}^{n} (\mathbf{u}_i^T \mathbf{x})^2.$$

***Proof*:** Omitted, please see [23] on pp.32.

The underlying assumption in LS problems is that error only occur in the vector $\mathbf{x}$ and the matrix H is exactly known, which is exactly the assumption we made earlier. In fact, this assumption is not uncommon and even with the consideration of the error in variables model (TLS), sometimes we still effectively are solving the LS problem indeed: consider the set $H\mathbf{s} \approx \mathbf{x}$, while the LS minimizes the sum of squared residuals $\|H\mathbf{s} - \mathbf{x}\|_2^2$, TLS in fact minimizes a sum of weighted squared residuals $\|\Delta H;\ \Delta\mathbf{x}\|_F$. In signal processing applications, if the signal $\mathbf{s}$ is unitary, as proved by Arun in [2], the solution to this unitarily constrained TLS problem is the same as the constrained LS problem. Especially, if there is correlation among the errors, then the TLS solution may no longer yield optimal statistical estimators [14]. Recently, A new detection technique called the maximum likelihood sphere decoding algorithm based on integer least square problem with QR decomposition was proposed in [47] to lower the computational complexity of the signal recovery problem, and consequently has draw a wide interests as a promising soft iterative decoding technique to improve the bit error rate performance of various communication

systems. In the following section, we will first explain the sphere decoding algorithm and then the maximum likelihood concepts.

2.3 Integer Least squares and Sphere Decoding

Now, consider the integer least-squares problem:

$$\min_{\mathbf{s}\in\mathcal{D}\subset\mathbb{Z}^m} \|\mathbf{x} - H\mathbf{s}\|_2 \tag{2.2}$$

where $\mathbf{x} \in \mathbb{R}^n$, $H \in \mathbb{R}^{n\times m}$, and $\mathbb{Z}^m$ denotes the m-dimensional integer lattice and $\mathcal{D} \subset \mathbb{Z}^m$, i.e., $\mathbf{s}$ is an m-dimensional vector with integer entries and the search space is a finite subset of the infinite lattice.

The integer least-squares problem has a simple geometric interpretation. As the entries of $\mathbf{s}$ run over the integers, $\mathbf{s}$ spans the "rectangular" m-dimensional lattice $\mathbb{Z}^m$. However, for any given *lattice generating matrix* H, the n-dimensional vector $H\mathbf{s}$ spans a "skewed" lattice (when $n > m$,this skewed lattice lives in an m-dimensional subspace of $\mathbb{R}^n$). Therefore, given the skewed lattice $H\mathbf{s}$, the integer least-squares problem is to find the "closest" lattice point $H\mathbf{s} \in \mathbb{R}^n$(in a Euclidean sense) to $\mathbf{x}$ — see Figure (2.1) below:

In sphere decoding, we attempt to search over only lattice points that lie in a certain hypersphere of radius ρ around the given vector $\mathbf{x}$ to reduce the search space and the required computations (see Figure (2.2)):

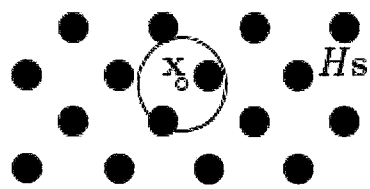

Figure 2.1: Geometrical interpretation of the integer least-square problem

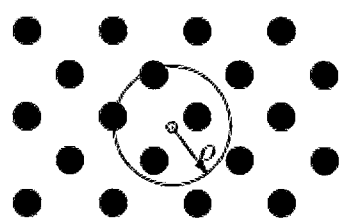

Figure 2.2: Idea behind the sphere decoder

However, two key questions need to be addressed here:

1. The choice of ρ. Clearly, if ρ is too large, we could obtain too many points and the search remains exponential in size, if ρ is too small, we could obtain no points inside the hyper-sphere.

2. Find the lattice points that are inside the hyper-sphere.

Sphere decoding does not really address the first question, but does propose an efficient way to answer the second. The basic observation is the following: Although it is difficult to determine the lattice points inside a general m-dimensional hyper-sphere, it is trivial to do so in one-dimensional case of $m = 1$. As one-dimensional hyper-sphere is simply an interval, so the desired lattice points will be the integer values that lie in this interval. We can use this observation to go from dimension k to $k + 1$: Suppose we have determined all k-dimensional lattice points that lie in a hyper-sphere of radius ρ, then for any such k-dimensional point, the set of admissible values of the $k + 1$-th dimensional coordinate that lie in the higher dimensional sphere of the *same* radius ρ forms an interval.

The above means that we can determine all lattice points in a hyper-sphere of dimensional m and radius ρ by successively determining all lattice points in hyper-spheres of lower dimensions $1, 2, ..., m$ and the same radius ρ.

Now, assuming $n \geq m$, i.e. there are at least as many equations as unknowns in $\mathbf{x} \approx H\mathbf{s}$. Note that the lattice points $H\mathbf{s}$ lies in a hyper-sphere

of radius ρ iff

$$\rho^2 \geq \|\mathbf{x} - H\mathbf{s}\|^2 \tag{2.3}$$

In order to break the problem into the subproblems described above, it is useful to consider the QR factorization of H:

$$H = Q \begin{bmatrix} R \\ 0_{(n-m)\times m} \end{bmatrix} \tag{2.4}$$

where R is an $m \times m$ upper triangular matrix and $Q = [Q_1 \quad Q_2]$ is an $n \times n$ orthogonal matrix. The matrices Q_1, Q_2 represent the first m and last $n - m$ orthonormal columns of Q respectively. The condition (2.3) then can be written as(we will only discuss the matrix in $\mathbb{R}^n$):

$$\rho^2 \geq \left\|\mathbf{x} - [Q_1 \quad Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix}\mathbf{s}\right\|^2 = \left\|\begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}\mathbf{x} - \begin{bmatrix} R \\ 0 \end{bmatrix}\mathbf{s}\right\|^2 = \|Q_1^T\mathbf{x} - R\mathbf{s}\|^2 + \|Q_2^T\mathbf{x}\|^2$$

or:

$$\rho^2 - \|Q_2^T\mathbf{x}\|^2 \geq \|Q_1^T\mathbf{x} - R\mathbf{s}\|^2 \tag{2.5}$$

Defining $y = Q_1^T\mathbf{x}$ and $\widehat{\rho}^2 = \rho^2 - \|Q_2^T\mathbf{x}\|^2$, we have:

$$\widehat{\rho}^2 \geq \sum_{i=1}^{m}\left(y_i - \sum_{j=i}^{m} r_{ij}s_j\right)^2 \tag{2.6}$$

Since R is upper triangular, the RHS of (2.6) can be expanded as:

$$\widehat{\rho}^2 \geq (y_m - r_{mm}s_m)^2 + (y_{m-1} - r_{m-1,m}s_m - r_{m-1,m-1}s_{m-1})^2 + \dots \qquad (2.7)$$

where the first term depends only on s_m, the second term on s_m, s_{m-1} and so on. Therefore a necessary condition for $H\mathbf{s}$ to lie inside the hyper-sphere is that $\widehat{\rho}^2 \geq (y_m - r_{mm}s_m)^2$. This is equivalent to(Notice that by assumption, $\widehat{\rho} \geq 0$):

$$\left\lceil \frac{-\widehat{\rho} + y_m}{r_{mm}} \right\rceil \leq s_m \leq \left\lfloor \frac{\widehat{\rho} + y_m}{r_{mm}} \right\rfloor \qquad \texttt{if } r_{mm} > 0; \qquad (2.8)$$

or:

$$\left\lceil \frac{\widehat{\rho} + y_m}{r_{mm}} \right\rceil \leq s_m \leq \left\lfloor \frac{-\widehat{\rho} + y_m}{r_{mm}} \right\rfloor \qquad \texttt{if } r_{mm} < 0; \qquad (2.9)$$

For any s_m satisfying (2.8) or (2.9), defining $\widehat{\rho}^2_{m-1} = \widehat{\rho}^2 - (y_m - r_{mm}s_m)^2$ and $y_{m-1|m} = y_{m-1} - r_{m-1,m}s_m$, a strong necessary condition can be found by looking at the first two terms in (2.7), which leads to:

$$\left\lceil \frac{-\widehat{\rho}_{m-1} + y_{m-1|m}}{r_{m-1,m-1}} \right\rceil \leq s_{m-1} \leq \left\lfloor \frac{\widehat{\rho}_{m-1} + y_{m-1|m}}{r_{m-1,m-1}} \right\rfloor \qquad \texttt{if } r_{m-1,m-1} > 0;$$

or

$$\left\lceil \frac{\widehat{\rho}_{m-1} + y_{m-1|m}}{r_{m-1,m-1}} \right\rceil \leq s_{m-1} \leq \left\lfloor \frac{-\widehat{\rho}_{m-1} + y_{m-1|m}}{r_{m-1,m-1}} \right\rfloor \qquad \texttt{if } r_{m-1,m-1} < 0;$$

Continue this fashion for s_{m-2}, and so on until s_1, we can obtain all lattice points belonging to (2.3).

Thus, we can formalize the implementation of the sphere decoding algorithm as:

Input: $Q = [Q_1 \quad Q_2]$, R, $\mathbf{x}$, $\mathbf{y} = Q_1^T\mathbf{x}$, ρ.

1. Set $k = m$, $\widehat{\rho}_k^2 = \rho^2 - \|Q_2^T\mathbf{x}\|^2$, $y_{k|k+1} = y_k$

2. (Bounds for s_k)

   if $r_{k,k} > 0$, Set $UB(s_k) = \lfloor \frac{\widehat{\rho}_k + y_{k|k+1}}{r_{k,k}} \rfloor$, $s_k = \lceil \frac{-\widehat{\rho}_k + y_{k|k+1}}{r_{k,k}} \rceil - 1$.

   else, Set $UB(s_k) = \lfloor \frac{-\widehat{\rho}_k + y_{k|k+1}}{r_{k,k}} \rfloor$, $s_k = \lceil \frac{\widehat{\rho}_k + y_{k|k+1}}{r_{k,k}} \rceil - 1$.

3. (Increase s_k) $s_k = s_k + 1$, If $s_k \leq UB(s_k)$, go to 5, else go to step 4.

4. (Increase k) $k = k + 1$, if $k > m$, return results and terminate, else go to 3.

5. If $k = 1$, go to next step. Else, save s_k, and set $k = k - 1$, $y_{k|k+1} = y_k - \sum_{j=k+1}^{m} r_{kj}s_j$, $\widehat{\rho}_k^2 = \widehat{\rho}_{k+1}^2 - \left(y_{k+1|k+2} - r_{k+1,k+1}s_{k+1}\right)^2$ and go to 2.

6. Solution found. Save s_k and go to 3.

2.4 Maximum likelihood and Sphere Decoding

Among all the decoding methods, the maximum likelihood decoder yields the best performance. The performances of Space-Time Codes for a large number

of antennas is studied using three different decoders: ML (Maximum Likelihood), MMSE (Mimimum Mean Square Error) and ZF (Zero Forcing) in [7]. The maximum likelihood decoder computes the estimate codeword $\hat{\mathbf{x}}$ by

$$\hat{\mathbf{x}} = min_{\mathbf{x}\in D}\|\mathbf{x} - H\mathbf{s}\|_2^2$$

where D as defined in (2.2), $\mathbf{s}$ and $\mathbf{x}$ are the transmitted and received signals respectively, and $\hat{\mathbf{x}}$ is searched over all possible $\mathbf{x}$ in the lattice space to minimize the 2-norm using the statistical property of the noise.

For BPSK (binary phase shift keying), the D in (2.2) is 2, so the decoder has to perform an enumeration over a set size 2^M. For higher-order modulation such as 64-QAM this complexity can become prohibitive even for a small number of transmit antennas. Thus, using ML method to choose an appropriate searching radius is necessary to reduce the computational complexity.

In our case, to answer the second question we just asked, the initial radius must be big enough to ensure that at least one lattice point is inside the sphere. One method is to use statistical properties of the channel noise. Notice that $\|b\|^2/\sigma^2$ has a Chi-Square distribution [26] with $2NL$ degree of freedom, where N is the number of receivers and L is the channel order. The probability of the lattice point inside the sphere can be written as

$$Pr[\|b\|^2 \leq \rho^2] = Pr[\frac{\|b\|^2}{\sigma^2} \leq \frac{\|\rho\|^2}{\sigma^2}]$$
$$= \int_0^{\rho^2/\sigma^2} \frac{\lambda^{NL-1}}{2^{NL}\Gamma(NL)e^{-\lambda/2}d\lambda},$$

where

$$\Gamma(NL) = \int_0^{\infty} t^{NL-1}e^{-t}dt.$$

One possible choice of the radius ρ is

$$\rho^2 = 2\alpha NL\sigma^2, \tag{2.10}$$

and $\alpha \geq 1$ is chosen so that we can be sure there exist a solution inside the sphere. Afterall, if the point is not found, we can always increase the probability of Pr to adjust the radius and search again. The complexity of the sphere decoding algorithm has been shown in [13, 21] to be the order of

$$O\left(n^2 \times (1 + \frac{n-1}{4\lambda^{-1}\rho^2})^{4\lambda^{-1}\rho^2}\right),$$

where n is the lattice dimension, ρ is the radius of the sphere, and λ is the lower bound for the eigenvalues of the matrix RR^T.

Using such statistical property of the channel noise lead to the necessary choice of the searching radius in our sphere decoding algorithm, and the combination of the two makes the name *maximum likelihood sphere decoding.*

However, notice that we are searching the solutions in the lattice space, it is natural to ask if we can use the lattice properties in the sphere decoding. After all, the sphere decoding itself was proposed with the name of universal lattice code decoder beared in [47]. One of the most important concepts in lattice theory is the basis transformations, or the basis reductions, among these the so called *lattice reduction aided* detection [49, 48] has shown a very promising result: the bit error rate (BER) parallel those for maximum likelihood detection. While a basis change does not always lead to an optimum perfomance, a change towards more orthogonal basses in general improves the performance. The more correlated the columns of H are, the more significant the improvement in performance is [4]. In this thesis, we will see that the searching radius in the sphere decoding can be decided using the reduction algorithm instead of the Gaussian probability function. So the following we will first introduce the lattice basis reduction with focus on the most celebrated one - LLL basis reduction.

Chapter 3

Introduction to the LLL algorithm

The lattice reduction has recently been rediscovered as a numerical method in many applications including MIMO systems and cryptography. In many cases, a basis consisting of relatively short and nearly orthogonal vectors is desirable, we will examine an important lattice basis reduction algorithm–the Lenstra, Lenstra and Lovasz (LLL) algorithm, and show one of its important usage in breaking the Merkle-Hellman code.

3.1 Lattices and Basis Reduction

In this section, we will briefly introduce some of the basic lattice concepts.

Although there are a couple of handful definitions for lattice, the one

in [31] captures the essence of the lattices.

Definition 1 *Let n be a positive integer. A subset L of the n-dimensional real vector space $\mathbb{R}^n$ is called a* lattice *if there exists a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ of $\mathbb{R}^n$ such that*

$$L = \sum_{i=1}^{n} \mathbb{Z}\mathbf{b}_i = \{\sum_{i=1}^{n} r_i \mathbf{b}_i : r_i \in \mathbb{Z}(1 \leq i \leq n)\}$$

Furthermore, we say that $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ form a *basis* for L.

The basis is not uniquely determined. To prove this, we need to introduce the unimodular concept. Series of elementary column operations can be described by so-called unimodular matrices [44].

Definition 2 *A nonsingular matrix M is unimodular if M is integral and $det(M) = \pm 1$.*

Lemma 3.1.1 *A nonsingular integer matrix M is unimodular if and only if M^{-1} is an integer matrix.*

We further give a theorem and two corollaries here to establish the relations between the unimodular matrix and lattice generating matrix. They can also be found in [44].

Theorem 3.1.1 *(Hermite normal form theorem). Each rational matrix of full rank can be brought into Hermite normal form by a series of elementary column operations.*

corollary 3.1.1 *Let A and A' be nonsingular matrices. Then the following are equivalent:*

1. *the columns of A and those of A' generate the same lattice;*
2. *A' comes from A by elementary column operations;*
3. *$A' = AM$ for some unimodular matrix M (i.e. $A^{-1}A'$ is unimodular).*

corollary 3.1.2 *For each rational matrix A of full rank there is a unimodular matrix M such that AM is the Hermite normal form of A. If A is nonsingular, M is unique.*

Now, let M be an $n \times n$ unimodular nonsingular integer matrix and B be a basis for a lattice L, then $L = B\mathbb{Z}^n$ and $BM\mathbb{Z}^n$ is necessarily a sub-lattice of $B\mathbb{Z}^n$ because $M\mathbb{Z}^n \subset \mathbb{Z}^n$, so $BM\mathbb{Z}^n \subset B\mathbb{Z}^n$. But M^{-1} also have all integer entries and $det(M^{-1}) = 1$, so $M^{-1}\mathbb{Z}^n \subset \mathbb{Z}^n$, that is, $\mathbb{Z}^n \subset M\mathbb{Z}^n$, so we have $B\mathbb{Z}^n \subset BM\mathbb{Z}^n$. Therefore, we can conclude that $B\mathbb{Z}^n = BM\mathbb{Z}^n$, i.e. BM is also a basis for $L = B\mathbb{Z}^n$.

Intuitively, for a 2-dimensional lattice, we can describe it as a set of intersection points of a regular (but not necessarily orthogonal) 2-dimensional infinite grid as follows.

The same lattice from Figure 3.1 can also be represented by the basis in Figure 3.2.

However, the determinant $d(L)$ of L defined by

$$d(L) = |det(b_1, b_2, \ldots, b_n)|, \tag{3.1}$$

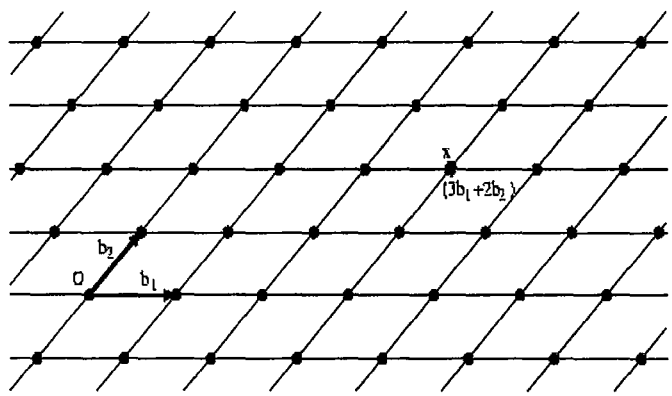

Figure 3.1: A Lattice in 2 dimensions

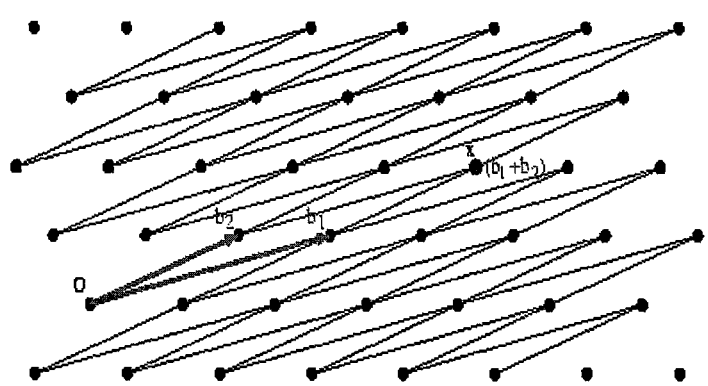

Figure 3.2: A different basis for the same lattice

the b_i being written as column vectors, is a positive real number that does not depend on the choice of the basis[8]. For if $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b}_n$ and $\mathbf{b_1}', \mathbf{b_2}', \ldots, \mathbf{b}_n'$ are bases of the same lattice, then by the above discussion,

$$det(\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b}_n) = det(\mathbf{b_1}', \mathbf{b_2}', \ldots, \mathbf{b}_n')det(M) = \pm det(\mathbf{b_1}', \mathbf{b_2}', \ldots, \mathbf{b}_n')$$

Hence,

$$d(L) = |det(\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b}_n)|$$

is independent of the particular choice of basis for L.

Next, we observe that given a lattice L with $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ as a basis and $\bar{L}$ is obtained by projecting L perpendicular to $\mathbf{b_1}$, then for any vector $\bar{\mathbf{v}} \in \bar{L}$, there is a unique vector $\mathbf{v}$, such that $-\frac{\|\mathbf{b_1}\|}{2} < (\mathbf{v}, \frac{\mathbf{b_1}}{\|\mathbf{b_1}\|}) \leq \frac{\|\mathbf{b_1}\|}{2}$ and $\bar{\mathbf{v}}$ is $\mathbf{v}$'s projection perpendicular to $\mathbf{b_1}$, where $||.||$ denotes the norm or the ordinary Euclidean length [25]. To see this, we have to understand two things:

First, the geometric meaning of dot product. From the Law of Cosines, we can derive geometric formula for dot product of two vectors $\mathbf{a}$ and $\mathbf{b}$ as

$$\mathbf{ab} = \|\mathbf{a}\|\|\mathbf{b}\| cos\theta,$$

where θ is the angle between $\mathbf{a}$ and $\mathbf{b}$. If we rewrite this as

$$\mathbf{a}(\frac{\mathbf{b}}{\|\mathbf{b}\|}) = \|\mathbf{a}\| cos\theta,$$

then we can interpret it as "the dot product of a vector with a unit vector is the scalar of that vector's projection on the unit vector's direction".

Thus the above observation says that for any vector $\bar{\mathbf{v}} \in \bar{L}$, there is a unique vector $\mathbf{v}$, such that its projection on $\mathbf{b_1}$ has magnitude no longer than half of $\mathbf{b_1}$'s.

Second, the basic facts about lattice. For example, in a 2-dimension lattice (see figure 1.3), let $\mathbf{b_1}$ be a basis and $\mathbf{v}$ is any vector in L, then the projection of $\mathbf{v}$, call it $\mathbf{v}_\perp$, is perpendicular to $\mathbf{b_1}$ and is equivalent to project $\hat{\mathbf{v}}$ onto $\mathbf{v}_\perp$. Furthermore, $\|\hat{\mathbf{v}} - \mathbf{v}_\perp\| \leq \frac{\|\mathbf{b_1}\|}{2}$ as $\mathbf{v}_\perp$ has to fall in between $\mathbf{v}'$ and $\hat{\mathbf{v}}$, the adjacent two lattice points. Since $\hat{\mathbf{v}} - \mathbf{v}_\perp$ is in fact the projection of $\hat{\mathbf{v}}$ onto $\mathbf{b_1}$ with magnitude of $\|\hat{\mathbf{v}}\| cos\theta$, which is exactly the dot product $(\hat{\mathbf{v}}, \frac{\mathbf{b_1}}{\|\mathbf{b_1}\|})$, so either $\mathbf{v}'$ or $\hat{\mathbf{v}}$ is the closest vector in L except when $\mathbf{v}_\perp$ exactly falling in the middle. Thus, if we choose the value interval to be $\left(-\frac{\|\mathbf{b_1}\|}{2}, \frac{\|\mathbf{b_1}\|}{2}\right]$, then the choice of such $\hat{\mathbf{v}}$ is unique and the observation follows.

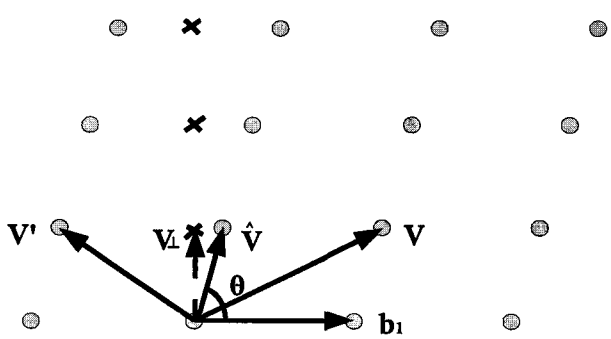

Figure 3.3: Vector's Projection in 2D Lattice

Finally, notice that $\mathbf{b_1}$ has to be a basis instead of any vector, if not, then the uniqueness does not follow. It could be more clear if we draw an example in 3-dimensional space, but let us skip it for the purpose of saving the space. By doing the above projecting and "unprojecting", we are guaranteed to have a new basis that is no longer than the original basis.

3.2 The LLL Basis Reduction and Algorithm

With the above knowledge equipped, we now can move on to the LLL algorithm now. The LLL algorithm due to A.K. Lenstra, H.W. Lenstra and L. Lovasz was proposed in 1982 [31]. It has been a foundation for many other improved algorithms since then. In this section, we will first introduce the LLL basis reduction and the properties of such reduced basis, then algorithm itself and some recent advances.

3.2.1 The LLL Reduced Lattice Basis and its Properties

Let $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n} \in \mathbb{R}^n$ be linearly independent, recall the Gram-Schmidt orthogonalization process:

$$\mathbf{b_i^*} = \mathbf{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b_j^*} \quad \textit{and} \tag{3.2}$$

$$\mu_{ij} = (\mathbf{b_i}, \mathbf{b_j^*})/\|\mathbf{b_j^*}\|^2, \tag{3.3}$$

where $1 \leq i \leq n$, the vectors $\mathbf{b_i^*}$ are orthogonal and depend on the ordering of $\mathbf{b_i}$'s.

Definition 3 *An ordered basis* $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ *for a lattice* L *is called LLL reduced if*

$$|\mu_{ij}| \leq 1/2 \quad for \quad 1 \leq j < i \leq n \tag{3.4}$$

and

$$\|\mathbf{b_i^*} + \mu_{i(i-1)}\mathbf{b_{i-1}^*}\|^2 \geq \frac{3}{4}\|\mathbf{b_{i-1}^*}\|^2 \quad for \quad 1 < i \leq n, \tag{3.5}$$

where $\mathbf{b_i^*}$ *and* μ_{ij} *are defined as in (3.2) and (3.3) respectively.*

We now give some properties of a basis that satisfies these conditions and explain their geometric meaning along the way.

Proposition 3.2.1 *Let* $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ *be a reduced basis for a lattice* L *in* $\mathbb{R}^n$ *as in Definition* 3, *and* $\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_n^*}$ *be defined as in (3.2) and (3.3). Then the following hold:*

$$\|\mathbf{b_j}\|^2 \leq 2^{i-1} \cdot \|\mathbf{b_i^*}\|^2, \tag{3.6}$$

$$d(L) \leq \Pi_{i=1}^{n}\|\mathbf{b_i}\| \leq 2^{n(n-1)/4} \cdot d(L), \tag{3.7}$$

$$\|\mathbf{b_1}\| \leq 2^{(n-1)/4} \cdot d(L)^{1/n}. \tag{3.8}$$

Notice that $\frac{3}{4}$ in (3.5) is arbitrarily chosen and could be replaced by $y \in (\frac{1}{4}, 1)$, subsequently the power of 2 appearing in (3.6), (3.7) and (3.8) must be replaced by same powers of $4/(4y-1)$.

Proof: Since

$$\|\mathbf{b}_i^* + \mu_{i(i-1)}\mathbf{b}_{i-1}^*\|^2 = \|\mathbf{b}_i^*\|^2 + \|\mu_{i(i-1)}\mathbf{b}_{i-1}^*\|^2 \quad for\ \mathbf{b}_i^* \perp \mathbf{b}_{i-1}^*,$$

so from (3.5) and (3.4), we have

$$\|\mathbf{b}_i^*\|^2 \geq (\frac{3}{4} - \mu_{i(i-1)})\|\mathbf{b}_{i-1}^*\|^2 \geq \frac{1}{2}\|\mathbf{b}_{i-1}^*\|^2$$

for $1 < i \leq n$. Hence by induction $\|\mathbf{b}_j^*\|^2 \leq 2^{i-j} \cdot \|\mathbf{b}_i^*\|^2$ for $1 \leq j \leq i \leq n$. Further from (3.2) and again (3.4), we have

$$\begin{aligned}
\|\mathbf{b}_i\|^2 &= \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1} \mu_{ij}^2 \|\mathbf{b}_j^*\|^2 \\
&\leq \|\mathbf{b}_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j} \|\mathbf{b}_i^*\|^2 \\
&= (1 + \frac{1}{4}(2^i - 2)) \cdot \|\mathbf{b}_i^*\|^2 \\
&\leq 2^{i-1} \cdot \|\mathbf{b}_i^*\|^2.
\end{aligned}$$

It follows that $\|\mathbf{b}_j\|^2 \leq 2^{j-1} \cdot \|\mathbf{b}_j^*\|^2 \leq 2^{i-1} \cdot \|\mathbf{b}_i^*\|^2$. This proves (3.6).

By *Hadamard's* inequality (see [9] on Page 81), $d(L) \leq \Pi_{i=1}^{n}\|\mathbf{b_i}\|$ is trivial. And from above we have $\|\mathbf{b_i}\| \leq 2^{(i-1)/2} \cdot \|\mathbf{b_i^*}\|$, together with the fact that $\mathbf{b_i^*}$s are pairwise orthogonal, which gives $d(L) = \Pi_{i=1}^{n}\|\mathbf{b_i^*}\|$, we conclude

$$\begin{aligned}
\Pi_{i=1}^{n}\|\mathbf{b_i}\| &\leq \Pi_{i=1}^{n} 2^{(i-1)/2}\|\mathbf{b_i^*}\| \\
&= 2^{\sum_{i=1}^{n}(\frac{i-1}{2})}\Pi_{i=1}^{n}\|\mathbf{b_i^*}\| \\
&= 2^{\frac{1}{2}\sum_{i=1}^{n-1} i} d(L) \\
&= 2^{\frac{1}{2}\frac{n(n-1)}{2}} d(L) \\
&= 2^{\frac{n(n-1)}{4}} d(L).
\end{aligned}$$

This proves (3.7).

Finally from (3.6), putting $j = 1$ yields

$$\|\mathbf{b_1}\| \leq 2^{(i-1)/2} \cdot \|\mathbf{b_i^*}\|, \quad for \quad 1 \leq i \leq n.$$

That is,

$$\begin{aligned}
&\underbrace{\|\mathbf{b_1}\| \cdot \|\mathbf{b_1}\| \cdots \|\mathbf{b_1}\|}_{n} \leq \Pi_{i=1}^{n} 2^{(i-1)/2}\|\mathbf{b_i^*}\| \\
\Longrightarrow\ &\|\mathbf{b_1}\|^n \leq 2^{\sum_{i=1}^{n}(\frac{i-1}{2})}\Pi_{i=1}^{n}\|\mathbf{b_i^*}\|
\end{aligned}$$

$$\Longrightarrow \|\mathbf{b_1}\|^n \leq 2^{\frac{n(n-1)}{4}} d(L)$$

$$\Longrightarrow \|\mathbf{b_1}\| \leq 2^{(n-1)/4} d(L)^{1/n}$$

This proves (3.8). ■

Furthermore, the following properties also hold:

Proposition 3.2.2 *For every* $\mathbf{x} \in L$ *and* $\mathbf{x} \neq 0$, *where* $L \subset \mathbb{R}^n$ *is a lattice with LLL reduced basis* $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ *and* $\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_n^*}$ *defined as in 3.2, then*

$$\|\mathbf{b_1}\|^2 \leq 2^{n-1} \cdot \|\mathbf{x}\|^2. \tag{3.9}$$

Further, if $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t} \in L$ *are linear independent, where* $t \leq n$, *then*

$$\|\mathbf{b_j}\|^2 \leq 2^{n-1} max\{\|\mathbf{x_1}\|^2, \|\mathbf{x_2}\|^2, \ldots, \|\mathbf{x_t}\|^2\} \tag{3.10}$$

for $j = 1, 2, \ldots, t$.

***Proof*:** Since $\mathbf{x} \in L$, so $\mathbf{x} = \sum_{i=1}^{n} z_i \mathbf{b_i} = \sum_{i=1}^{n} r_i \mathbf{b_i^*}$, where $z_i \in \mathbb{Z}$, $r_i \in \mathbb{R}$ and $1 \leq i \leq n$.

From (3.2), we have $\mathbf{b_i} = \mathbf{b_i^*} + \sum_{j=1}^{i} \mu_{ij} \mathbf{b_j^*}$. That is,

$$\begin{aligned} x &= \sum_{i=1}^{n} z_i \mathbf{b_i} \\ &= \sum_{i=1}^{n} z_i (\mathbf{b_i^*} + \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b_j^*}) \end{aligned}$$

$$= \sum_{i=1}^{n} z_i \mathbf{b_i^*} + \sum_{i=1}^{n} z_i (\sum_{j=1}^{i-1} \mu_{ij} \mathbf{b_j^*})$$
$$= \sum_{i=1}^{n} r_i \mathbf{b_i^*},$$

so if i is the largest index such that $z_i \neq 0$, then $z_i = r_i$. Thus

$$\begin{aligned} \|\mathbf{x}\|^2 &\geq r_i^2 \cdot \|\mathbf{b_i^*}\|^2 \\ &= z_i^2 \cdot \|\mathbf{b_i^*}\|^2 \\ &\geq \|\mathbf{b_i^*}\|^2 \\ &\geq 2^{1-i} \|\mathbf{b_1}\|^2 \quad \textit{by (3.6)} \\ &\geq 2^{1-n} \|\mathbf{b_1}\|^2, \end{aligned}$$

which implies $\|\mathbf{b_1}\|^2 \leq 2^{n-1} \|\mathbf{x}\|^2$. This proves (3.9).

Again, write $\mathbf{x_j} = \sum_{i=1}^{n} z_{ij} \mathbf{b_i}$ with $z_{ij} \in \mathbb{Z}$ and $1 \leq i \leq n$, $1 \leq j \leq t$. For any fixed j, let $i(j)$ denote the largest i for which $z_{ij} \neq 0$. By the above proof, we have

$$\|\mathbf{x_j}\|^2 \geq \|\mathbf{b_{i(j)}^*}\|^2$$

for $1 \leq j \leq t$. Arrange the index $i(1) \leq i(2) \leq \cdots \leq i(t)$ for all $\mathbf{x_j}$'s, and renumber the correspondent $\mathbf{x_j}$'s as $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$. We claim that $j \leq i(j)$: if not, then by the above proof, the largest index $i(j) < j$, so $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_j}$ would all belong to $\mathbb{Z}\mathbf{b_1} + \mathbb{Z}\mathbf{b_2} + \cdots + \mathbb{Z}\mathbf{b_{j-1}}$, contradict to the fact that $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}$ are linearly independent.

Therefore,

$$\begin{aligned} 2^{n-1} \cdot \|\mathbf{x_j}\|^2 &\geq 2^{n-1} \cdot \|\mathbf{b}^*_{\mathbf{i(j)}}\|^2 \\ &\geq 2^{i(j)-1} \cdot \|\mathbf{b}^*_{\mathbf{i(j)}}\|^2 \\ &\geq \|\mathbf{b_j}\|^2 \quad \textit{by } (3.6) \end{aligned}$$

for $j = 1, 2, \ldots, t$. This proves 3.10.

Finally, let us investigate a bit further about the definition and its properties.

The LLL algorithm is actually a generalization of Gaussian reduction in n-dimensions. So it is natural that we first consider the Gaussian basis reduction in 2-dimensions.

Suppose $\mathbf{b}_1, \mathbf{b}_2$ forms the basis of a lattice $L \in \mathbb{R}^2$. Rename them so that $\|\mathbf{b}_1\| \leq \|\mathbf{b}_2\|$. Let $\mathbf{b}_2{}^* = \mathbf{b}_2 - \frac{(\mathbf{b}_1, \mathbf{b}_2)}{\|\mathbf{b}_1\|^2}\mathbf{b}_1$, clearly, integer t that is closest to $\frac{(\mathbf{b}_1, \mathbf{b}_2)}{\|\mathbf{b}_1\|^2}$ shortens $\mathbf{b}_2$ such that $\mathbf{b}_2{}' = \mathbf{b}_2 - t \cdot \mathbf{b}_1$, and the projection of $\mathbf{b}_2{}'$ onto $\mathbf{b}_1$ has length at most $\|\mathbf{b}_1\|/2$. Further, if $\|\mathbf{b}_2{}'\| \geq \|\mathbf{b}_1\|$ we stop, otherwise we swap them and repeat the procedure. This can be written in an algorithm as:

GaussianReduction($\mathbf{b}_1\mathbf{b}_2$)

do

if $\|\mathbf{b}_1\| > \|\mathbf{b}_2\|$

$\mathbf{b} = \mathbf{b}_2$; $\mathbf{b}_2 = \mathbf{b}_1$; $\mathbf{b}_1 = \mathbf{b}$ (swap $\mathbf{b}_1, \mathbf{b}_2$)

$t = \lfloor \frac{(\mathbf{b}_1 \cdot \mathbf{b}_2)}{\|\mathbf{b}_1\|^2} \rceil$

$\mathbf{b_2} = \mathbf{b_2} - t\mathbf{b_1}$

while $\|\mathbf{b_1}\| > \|\mathbf{b_2}\|$

return $(\mathbf{b_1}, \mathbf{b_2})$

Clearly, the Gaussian Reduction terminates in a finite amount of steps as there are only finite number of lattice points with length smaller than $\|\mathbf{b_1}\|$. Furthermore, we can formulate the conditions when the above algorithm finds the shortest vector as

$$\|\mathbf{b_1}\| \leq \|\mathbf{b_2}\|, \tag{3.11}$$

$$|\mu_{21}| \leq \frac{1}{2}. \tag{3.12}$$

It follows that at end of the procedure, the acute angle between $\mathbf{b_1}$ and $\mathbf{b_2}$ is at least 60 degrees, so they are "fairly" orthogonal and the basis is reduced.

Recall the Gram-Schmidt Orthogonalization given by $\mathbf{b_2} = \mu_{21}\mathbf{b_1^*} + \mathbf{b_2^*}$, we have

$$\|\mathbf{b_2}\|^2 = \|\mu_{21}\mathbf{b_1^*} + \mathbf{b_2^*}\|^2.$$

Together with (3.11) and $\mathbf{b_1} = \mathbf{b_1^*}$ we have

$$\|\mathbf{b_1}\|^2 \leq \mu_{21}^2\|\mathbf{b_1^*}\|^2 + \|\mathbf{b_2^*}\|^2 \quad \Longrightarrow \quad \|\mathbf{b_1^*}\|^2 \leq \mu_{21}^2\|\mathbf{b_1^*}\|^2 + \|\mathbf{b_2^*}\|^2.$$

If we slightly modify Gauss's algorithm by: if $\|\mathbf{b}_2'\| \geq (1-\epsilon)\mathbf{b}_1$ we stop, else we swap them and repeat the procedure, where ϵ is any positive constant, then the length of the shortest basis vector falls by a factor at least $(1-\epsilon)$ each iteration. That is, we replace the **while** condition with $\|\mathbf{b}_2 < (1-\epsilon)\mathbf{b}_1\|$. It is easy to prove that the number of iterations of the algorithm is $O(log_{(1-\epsilon)^{-1}}\|\mathbf{b}_1^{(0)}\|)$, where $\mathbf{b}_1^{(0)}$ is the initial $\mathbf{b}_1$. [1] Thus, the running-time is bounded above by a polynomial. In this case, the basis $\mathbf{b}_1$, $\mathbf{b}_2$ at the end satisfies

$$\|\mathbf{b}_2\| \geq (1-\epsilon)\|\mathbf{b}_1\| \tag{3.13}$$

$$\mu_{21} = \frac{(\mathbf{b}_2 \cdot \mathbf{b}_1)}{\|\mathbf{b}_1\|^2} \leq 1/2, \tag{3.14}$$

whence we also have

$$\begin{aligned}
&\|\mathbf{b}_2^*\|^2 = \|\mathbf{b}_2 - \frac{(\mathbf{b}_2 \cdot \mathbf{b}_1)}{\|\mathbf{b}_1\|^2}\mathbf{b}_1^*\|^2 \implies \\
&\|\mathbf{b}_2^*\|^2 \geq \|\mathbf{b}_2\|^2 - \frac{1}{4}\|\mathbf{b}_1\|^2 \quad (\textit{notice } \mathbf{b}_1 = \mathbf{b}_1^*) \implies \\
&\|\mathbf{b}_2^*\|^2 \geq \left((1-\epsilon)^2 - \frac{1}{4}\right)^2 \|\mathbf{b}_1\|^2 \implies \\
&\|\mathbf{b}_2^*\| \geq \sqrt{(1-\epsilon)^2 - \frac{1}{4}}\|\mathbf{b}_1\|.
\end{aligned}$$

[1] **proof:** Assume after k many steps we can finally reduce the length of $\mathbf{b}_2$ to a positive constant c and stop, then we must have $\alpha(1-\epsilon)^k \cdot \|\mathbf{b}_1^{(0)}\| \leq c$, where $\alpha < 1$, which implies $k \leq \frac{ln(\|\mathbf{b}_1{}^{(0)}\|/(\alpha c))}{ln(1-\epsilon)^{-1}}$. That is, $k \in O(log_{(1-\epsilon)^{-1}}\|\mathbf{b}_1^{(0)}\|)$.

We choose ϵ so that $(1-\epsilon)^2-\frac{1}{4} > 0$ and $0 < 1-\epsilon < 1$ yields $0 < \epsilon < 1/2$. That is,

$$\|\mathbf{b_2}\| \geq (1-\epsilon)\|\mathbf{b_1}\| \implies \|\mathbf{b_2^*} + \mu_{21}\mathbf{b_1^*}\| \geq (1-\epsilon)\|\mathbf{b_1^*}\|^2,$$

where $\frac{1}{4} < (1-\epsilon) < 1$.

Now, the lattice defined by a pair of vectors $\mathbf{b_i}$ and $\mathbf{b_{i+1}}$ has basis

$$\mathbf{b_{i-1}} = \mathbf{b_{i-1}^*} \quad \textit{and} \quad \mathbf{b_i} = \mu_{i(i-1)}\mathbf{b_{i-1}^*} + \mathbf{b_i^*},$$

where $\mu_{i(i-1)}\mathbf{b_{i-1}^*} + \mathbf{b_i^*}$ and $\mathbf{b_{i-1}^*}$ are the projections of $\mathbf{b_i}$ and $\mathbf{b_{i-1}}$ on the orthogonal complement of $\sum_{j=1}^{i-2}\mathbb{R}\mathbf{b_j}$. We can generalize this by translating the relaxed 2-dimensional Gaussian Reduction conditions (3.13) and (3.14) to apply pairwise to adjacent vectors in this basis and thus gives the LLL basis reduction conditions

$$|\mu_{ij}| \leq 1/2 \quad \textit{for} \quad 1 \leq j < i \leq n$$

and

$$\|\mathbf{b_i^*} + \mu_{i(i-1)}\mathbf{b_{i-1}^*}\|^2 \geq \delta\|\mathbf{b_{i-1}^*}\|^2 \quad \textit{for} \quad 1 < i \leq n,$$

where $\frac{1}{4} < \delta < 1$.

The propositions describe the properties of such reduced basis. In particular, **Proposition** (3.2.1) states the basis is reasonably orthogonal (by (3.7)) and vector $\mathbf{b_1}$ is a good approximation of the shortest vector (3.8) [2]. **Proposition** (3.2.2) further states that $\mathbf{b_1}$ is a good approximation of the shortest vector again by giving an upper bound in terms of any vector (3.9) and each of the reduced basis vector is a good approximation to the successive minima of l_2 norm of the lattice L.

Now, we have cleared up some concepts about LLL reduction and its properties. It is time that we write down the LLL algorithm and explore its applications in various areas.

3.2.2 The LLL Algorithm

To start with, we compute b_i^* $(1 \leq i \leq n)$ and μ_{ij} $(1 \leq j < i \leq n)$ using (3.2) and (3.3), and then update them during the algorithm based on the satisfaction of conditions (3.4) and (3.5). Perhaps the original paper is the best one that explains the algorithm, so here we will just present the original algorithm and give some necessary explanations.

The Original LLL Algorithm

[2] As compared to Minkowski's theorem: let the length of the shortest non-zero vector of a lattice $L \in \mathbb{R}^n$ be denoted $\lambda(L)$ and determinant as $d(L)$, then $\lambda(L) \leq \sqrt{n} \cdot d(L)^{1/n}$.

$$\left.\begin{array}{l} \mathbf{b}_i^* := \mathbf{b}_i; \\ \left.\begin{array}{l} \mu_{ij} := (\mathbf{b}_i, \mathbf{b}_j^*)/B_j; \\ \mathbf{b}_i^* := \mathbf{b}_i^* - \mu_{ij}\mathbf{b}_j^* \end{array}\right\} \quad for \quad j = 1, 2, \ldots, i-1; \\ B_i := (\mathbf{b}_i^*, \mathbf{b}_i^*) \end{array}\right\} for \quad i = 1, 2, \ldots, n;$$

$k := 2$

(1) *perform* $(*)$ *for* $l = k - 1$;

if $B_k < (\frac{3}{4} - \mu_{k(k-1)}^2)B_{k-1}$, *go to* **(2)**;

perform $(*)$ *for* $l = k-2, k-3, \ldots, 1$;

if $k = n$, *terminate*;

$k := k + 1$;

go to **(1)**;

(2) $\mu := \mu_{k(k-1)}$; $B := B_k + \mu^2 B_{k-1}$; $\mu_{k(k-1)} := \mu B_{k-1}/B$;

$B_k := B_{k-1}B_k/B$; $B_{k-1} := B$;

$$\begin{pmatrix} b_{k-1} \\ b_k \end{pmatrix} := \begin{pmatrix} b_k \\ b_{k-1} \end{pmatrix};$$

$$\begin{pmatrix} \mu_{(k-1)j} \\ \mu_{kj} \end{pmatrix} := \begin{pmatrix} \mu_{kj} \\ \mu_{(k-1)j} \end{pmatrix}; \quad for\ j = 1, 2, \ldots, k-2;$$

$$\begin{pmatrix} \mu_{i(k-1)} \\ \mu_{ik} \end{pmatrix} := \begin{pmatrix} 1 & \mu_{k(k-1)} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & -\mu \end{pmatrix} \begin{pmatrix} \mu_{i(k-1)} \\ \mu_{ik} \end{pmatrix}; \quad for\ i = k+1, k+2, \ldots, n;$$

if $k > 2$, *then* $k := k - 1$;

go to **(1)**.

$(*)$ *if* $|\mu_{kl}| > \frac{1}{2}$, *then* :

$$\left\{\begin{array}{ll} r := \lfloor \mu_{kl} \rceil; & b_k := b_k - r b_l; \\ \mu_{kj} := \mu_{kj} - r\mu_{lj} & for\ j = 1, 2, \ldots, l-1; \\ \mu_{kl} := \mu_{kl} - r. & \end{array}\right.$$

Now, we explain how we get **(2)** in the algorithm.

Let $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$ be the current basis and $\mathbf{b_i^*}, \mu_{ij}$ as in (3.2) and (3.3). Let k be the current subscript for which the followings hold:

$$|\mu_{ij}| \leq \frac{1}{2} \quad for\ 1 \leq j < i < k, \tag{3.15}$$

$$\|\mathbf{b_i^*} + \mu_{i(i-1)}\mathbf{b_{i-1}^*}\|^2 \geq \frac{3}{4}\|\mathbf{b_{i-1}^*}\|^2 \quad for\ 1 < i < k, \tag{3.16}$$

and

$$|\mu_{k(k-1)}| \leq \frac{1}{2} \quad if\ k \geq 2, \tag{3.17}$$

$$\|\mathbf{b_k^*} + \mu_{k(k-1)}\mathbf{b_{k-1}^*}\|^2 < \frac{3}{4}\|\mathbf{b_{k-1}^*}\|^2 \quad if\ k \geq 2. \tag{3.18}$$

Denote the newly obtained basis as $\mathbf{c_i}$, $\mathbf{c_i^*}$ and v_{ij} that will replace $\mathbf{b_i}$, $\mathbf{b_i^*}$ and μ_{ij} respectively, then

$$\mathbf{c_i} = \mathbf{b_i} \quad for\ \ i \neq k-1, k,$$

and otherwise

$$\mathbf{c_{k-1}} = \mathbf{b_k},$$

$$\mathbf{c_k} = \mathbf{b_{k-1}}.$$

Since $\mathbf{c_{k-1}^*}$ is the projection of $\mathbf{c_{k-1}}$ on the orthogonal complement of

$\sum_{j=1}^{k-2} \mathbb{R}\mathbf{c_j}$, we have

$$\begin{aligned}
\mathbf{c}^*_{\mathbf{k-1}} &= \mathbf{c_{k-1}} - \sum_{j=1}^{k-2} \frac{(\mathbf{c_{k-1}}, \mathbf{c}^*_{\mathbf{j}})}{\|\mathbf{c}^*_{\mathbf{j}}\|^2} \mathbf{c}^*_{\mathbf{j}} \\
&= \mathbf{b_k} - \sum_{j=1}^{k-2} \frac{(\mathbf{b_k}, \mathbf{b}^*_{\mathbf{j}})}{\|\mathbf{b}^*_{\mathbf{j}}\|^2} \cdot \mathbf{b}^*_{\mathbf{j}} \\
&= \underbrace{\mathbf{b_k} - \frac{(\mathbf{b_k}, \mathbf{b}^*_{\mathbf{k-1}})}{\|\mathbf{b}^*_{\mathbf{k-1}}\|^2} \cdot \mathbf{b}^*_{\mathbf{k-1}} - \sum_{j=1}^{k-2} \frac{(\mathbf{b_k}, \mathbf{b}^*_{\mathbf{j}})}{\|\mathbf{b}^*_{\mathbf{j}}\|^2} \cdot \mathbf{b}^*_{\mathbf{j}}}_{\mathbf{b}^*_{\mathbf{k}}} + \frac{(\mathbf{b_k}, \mathbf{b}^*_{\mathbf{k-1}})}{\|\mathbf{b}^*_{\mathbf{k-1}}\|^2} \cdot \mathbf{b}^*_{\mathbf{k-1}} \\
&= \mathbf{b}^*_{\mathbf{k}} + \mu_{k(k-1)} \cdot \mathbf{b}^*_{\mathbf{k-1}}. \qquad (3.19)
\end{aligned}$$

Similarly, we also have

$$\begin{aligned}
v_{k(k-1)} &= (\mathbf{c_k}, \mathbf{c}^*_{\mathbf{k-1}}) / \|\mathbf{c}^*_{\mathbf{k-1}}\|^2 \\
&= (\mathbf{b_{k-1}}, \mathbf{b}^*_{\mathbf{k}} + \mu_{k(k-1)} \cdot \mathbf{b}^*_{\mathbf{k-1}}) / \|\mathbf{c}^*_{\mathbf{k-1}}\|^2 \\
&= (\mathbf{b}^*_{\mathbf{k-1}} + \sum_{j=1}^{k-2} \mu_{(k-1)j} \mathbf{b}^*_{\mathbf{j}}, \mathbf{b}^*_{\mathbf{k}} + \mu_{k(k-1)} \cdot \mathbf{b}^*_{\mathbf{k-1}}) / \|\mathbf{c}^*_{\mathbf{k-1}}\|^2 \\
&= \mu_{k(k-1)} \|\mathbf{b}^*_{\mathbf{k-1}}\|^2 / \|\mathbf{c}^*_{\mathbf{k-1}}\|^2 \qquad (3.20)
\end{aligned}$$

and

$$\mathbf{c}^*_{\mathbf{k}} = \mathbf{c_k} - \sum_{j=1}^{k-1} \frac{(\mathbf{c_k}, \mathbf{c}^*_{\mathbf{j}})}{\|\mathbf{c}^*_{\mathbf{j}}\|^2} \mathbf{c}^*_{\mathbf{j}}$$

$$
\begin{aligned}
&= \mathbf{c_k} - \sum_{j=1}^{k-2} \frac{(\mathbf{c_k}, \mathbf{c_j^*})}{\|\mathbf{c_j^*}\|^2}\mathbf{c_j^*} - \frac{(\mathbf{c_k}, \mathbf{c_{k-1}^*})}{\|\mathbf{c_{k-1}^*}\|^2}\mathbf{c_{k-1}^*} \\
&= \underbrace{\mathbf{b_{k-1}} - \sum_{j=1}^{k-2} \frac{(\mathbf{b_{k-1}}, \mathbf{b_j^*})}{\|\mathbf{b_j^*}\|^2}\mathbf{b_j^*}}_{\mathbf{b_{k-1}^*}} - \frac{(\mathbf{c_k}, \mathbf{c_{k-1}^*})}{\|\mathbf{c_{k-1}^*}\|^2}\mathbf{c_{k-1}^*} \\
&= \mathbf{b_{k-1}^*} - v_{k(k-1)}\mathbf{c_{k-1}^*}. \qquad (3.21)
\end{aligned}
$$

The interchange of the vectors $\mathbf{c_k}$ and $\mathbf{c_{k-1}}$ resulted the change of $\mathbf{c_k^*}$ and $\mathbf{c_{k-1}^*}$, and therefore the consequent $v_{i(k-1)}$ and v_{ik} for all $i > k$. [3] To find $v_{i(k-1)}$ and v_{ik}, we substitute

$$
\begin{aligned}
\mathbf{b_{k-1}^*} &= v_{k(k-1)}\mathbf{c_{k-1}^*} + \mathbf{c_k^*} \quad \textit{by (3.21)} \qquad (3.22) \\
\mathbf{b_k^*} &= (1 - \mu_{k(k-1)}v_{k(k-1)})\mathbf{c_{k-1}^*} - \mu_{k(k-1)}\mathbf{c_k^*} \quad \textit{by(3.19) and (3.22)} \\
&= (1 - \frac{\mu_{k(k-1)}^2\|\mathbf{b_{k-1}}\|^2}{\|\mathbf{c_{k-1}^*}\|^2})\mathbf{c_{k-1}^*} - \mu_{k(k-1)}\mathbf{c_k^*} \quad \textit{by (3.20)} \\
&= (\|\mathbf{b_k^*}\|^2/\|\mathbf{c_{k-1}^*}\|^2) \cdot \mathbf{c_{k-1}^*} - \mu_{k(k-1)}\mathbf{c_k^*} \quad \textit{by (3.19) again} \qquad (3.23)
\end{aligned}
$$

from above into $\mathbf{b_i} = \mathbf{b_i^*} + \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b_j^*}$. That yields

$$
\begin{aligned}
v_{i(k-1)} &= \mu_{i(k-1)}v_{k(k-1)} + \mu_{ik}\|\mathbf{b_k^*}\|^2/\|\mathbf{c_{k-1}^*}\|^2 \\
&= \mu_{i(k-1)}v_{k(k-1)} + \mu_{ik}(1 - \mu_{k(k-1)}v_{k(k-1)}) \quad \textit{by (3.23)} \qquad (3.24) \\
v_{ik} &= \mu_{i(k-1)} - \mu_{ik}\mu_{k(k-1)}. \qquad (3.25)
\end{aligned}
$$

[3] Notice that by our assumption that (3.15) and (3.16) hold, so v_{jk} does not change for all $j < k - 1$.

Finally, for $1 \leq j < k-1$ we have

$$v_{(k-1)j} = \mu_{kj}, \qquad v_{kj} = \mu_{(k-1)j},$$

and for $1 \leq j < i \leq n$, where $\{i,j\} \cap \{k-1,k\} = \emptyset$, we have $v_{ij} = \mu_{ij}$. It might be easier if one thinks the relationship between $\mathbf{b_i}$ and $\mathbf{b_i^*}, \mu_{ij}$ as vector-matrix type, that is

$$B = B^* \left[\mu_{ij}\right]^T,$$

where $B^* = [\mathbf{b_1^*}, \mathbf{b_2^*}, \ldots, \mathbf{b_n^*}]$ and $[\mu_{ij}]$ is a $n \times n$ lower triangular matrix with all diagonal elements equal to 1 (to see this, notice that $\mathbf{b_i} = \mathbf{b_i^*} + \sum_{j=1}^{i-1} \mu_{ij}\mathbf{b_j^*}$). The rest of (**2**) is straightforward except B_k, which is indeed $\|\mathbf{c_k^*}\|^2$. That is:

$$\begin{aligned}
\|\mathbf{c_k^*}\|^2 &= \|\mathbf{b_{k-1}^*} - v_{k(k-1)} \cdot \mathbf{c_{k-1}^*}\|^2 \quad \textit{by (3.21)} \\
&= \|\mathbf{b_{k-1}^*}\|^2 - 2v_{k(k-1)}\mu_{k(k-1)}\|\mathbf{b_{k-1}^*}\|^2 + v_{k(k-1)}^2\|\mathbf{c_{k-1}^*}\|^2 \\
&= \|\mathbf{b_{k-1}^*}\|^2 - 2v_{k(k-1)}^2\|\mathbf{c_{k-1}^*}\|^2 + v_{k(k-1)}^2\|\mathbf{c_{k-1}^*}\|^2 \quad \textit{by (3.22)} \\
&= \|\mathbf{b_{k-1}^*}\|^2 - v_{k(k-1)}^2\|\mathbf{c_{k-1}^*}\|^2 \\
&= \|\mathbf{b_{k-1}^*}\|^2 - \mu_{k(k-1)}^2\|\mathbf{b_{k-1}^*}\|^4/\|\mathbf{c_{k-1}^*}\|^2 \quad \textit{by (3.20)} \\
&= \|\mathbf{b_{k-1}^*}\|^2 \cdot \frac{\|\mathbf{c_{k-1}^*}\|^2 - \mu_{k(k-1)}^2\|\mathbf{b_{k-1}^*}\|^2}{\|\mathbf{c_{k-1}^*}\|^2} \\
&= B_{k-1} \cdot \frac{B_k + \mu_{k(k-1)}^2 B_{k-1} - \mu_{k(k-1)}^2 B_{k-1}}{B}
\end{aligned}$$

$$by\ (3.19)\quad B = \mathbf{c}_{\mathbf{k-1}}^{*} \implies B = B_k + \mu_{k(k-1)}B_{k-1}$$

$$= B_{k-1} \cdot B_k/B.$$

The beauty of the LLL algorithm is that it approximates the shortest vector in n-dimensional lattice in polynomial time while the other reductions such as Minkowski and Korkin-Zoloterref reductions are computationally too heavy [24]. In the following, we will further investigate some of the improvements and applications in different area with focus on wireless communication systems.

3.2.3 The LLL Algorithm's Improvements

An immediate improvement about the original LLL algorithm can be made if we use a variable k_{max} to keep the maximal value of k that has been obtained and compute the Gram-Schmidt coefficients as needed in stead of computing all the Gram-Schmidt coefficients μ_{kj} and B_k, and then updating μ_{ik} and $\mu_{i(k-1)}$ for $i > k$ after exchange of $\mathbf{b_k}$ and $\mathbf{b_{k-1}}$ (if there is any). After all, we know for sure that all $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_{k-2}}$ are LLL reduced.

This leads to an improved algorithm.

A Slightly Improved LLL Algorithm

Input: a lattice basis $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$.

Output: A LLL reduced basis $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$.

1. Set $k = 2$, $k_{max} = 1$, $\mathbf{b_1^*} = \mathbf{b_1}$ and $B_1 = \|\mathbf{b_1}\|_2$.

2. if $k \leq k_{max}$

   go to step 3.

   else

   set $k_{max} = k$, $\mathbf{b_k^*} = \mathbf{b_k}$;

   for $j = 1, \ldots, k-1$

   set $\mu_{kj} = \mathbf{b_k} \cdot \mathbf{b_j^*}/B_j$;

   $\mathbf{b_k^*} = \mathbf{b_k^*} - \mu_{kj}\mathbf{b_j^*}$.

   set $B_k = \|\mathbf{b_k^*}\|_2$.

3. Execute sub_algorithm **Reduce**$(k, k-1)$.

   if $B_k < (\frac{3}{4} - \mu_{k(k-1)}^2)B_{k-1}$

   execute sub_algorithm **Swap**(k);

   set $k = max(2, k-1)$ and go to step 3.

   else

   for $l = k-2, k-3, \ldots, 1$

   execute sub_algorithm **Reduce**(k, l);

   set $k = k + 1$.

4. if $k \leq n$

   go to step 2.

else

output the LLL reduced basis $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}$.

☞ **sub_algorithm Reduce**(k, l)

if $|\mu_{kl}| > \frac{1}{2}$

$r = \lfloor \mu_{k,l} \rceil$;

set $\mathbf{b_k} = \mathbf{b_k} - r \cdot \mathbf{b_l}$ and $\mu_{kl} = \mu_{kl} - r$.

for $j = 1, 2, \ldots, l-1$

$\mu_{kj} = \mu_{kj} - r \cdot \mu_{lj}$.

☞ **sub_algorithm Swap**(k)

exchange $\mathbf{b_k}$ and $\mathbf{b_{k-1}}$.

if $k > 2$

for $1 \leq j \leq k-2$

exchange μ_{kj} with $\mu_{(k-1)j}$.

set $\mu = \mu_{k(k-1)}$, $B = B_k + \mu^2 B_{k-1}$;

$\mu_{k(k-1)} = \mu B_{k-1}/B$, $B_k = B_{k-1} B_k / B$ and $B_{k-1} = B$.

for $i = k+1, k+2, \ldots, k_{max}$

set $t = \mu_{ik}$;

$\mu_{ik} = \mu_{i(k-1)} - \mu t$;

$$\mu_{i(k-1)} = t + \mu_{k(k-1)}\mu_{ik};$$ [4]

The μ_{ij}s and B_is are not necessarily rational numbers, in this case they must be represented approximately using floating point arithmetic. However, the round off errors in the floating point arithmetic may cause catastrophic divergence from the LLL algorithm and thus prevent the final basis to be LLL reduced (see [9] on Page 90). A variant of the LLL algorithm based on floating point arithmetic due to Schnorr was described in [40], it uses a so-called self-correction method to improve the accuracy of an initial approximation by one step of an iteration that converges quadratically to the correct values and the Schulz's method for approximating the inverse of a matrix. A practical floating point LLL algorithm that uses "deep insertions" and have good stability was proposed in 1993 by C.P. Schnorr and M. Euchner in [43], and the improvements latter by Henrik Koy and Claus P. Schnorr in 2001 [41, 28], recently by Nguyen and Stehle in 2005 [36] and Claus P. Schnorr in 2006 [42]. these improvements of LLL algorithm in floating point arithmetic made the LLL reduction algorithm efficiently applicable in practice.

[4] From (3.24) and (3.25) we know

$$\begin{aligned} v_{i(k-1)} &= \mu_{i(k-1)}v_{k(k-1)} + \mu_{ik}(1 - \mu_{k(k-1)}v_{k(k-1)}) \\ &= \mu_{ik} + (\mu_{i(k-1)} - \mu_{ik}\mu_{k(k-1)})v_{k(k-1)} \\ v_{ik} &= \mu_{i(k-1)} - \mu_{ik}\mu_{k(k-1)}, \end{aligned}$$

this leads to the iterative definitions in the algorithm.

3.3 The LLL Algorithm in Cryptography

The LLL algorithm has a broad practical uses in the cryptography. In this section, we will briefly introduce some basic ideas with focus on breaking the Merkle-Hellman code using the LLL algorithm.

3.3.1 Introduction to Ciphers

Public key cryptosystem, also known as asymmetric cryptosystem, was first introduced by W. Diffie and Martin Hellman in 1976 [12]. It has a pair of cryptographic keys – a public key and a private key. The private key is kept secret while the public key may be widely distributed. The keys are mathematically related, but the private key can not be practically derived from the public key. A message encrypted with the public key can be decrypted only with the corresponding private key. By contrast, the classical ciphers, as well as certain contemporary ciphers such as DES (Data Encryption Standard) and AES (Advanced Encryption Standard), are symmetric in the sense that knowledge of the decryption key is equivalent to, or often exactly equal to, knowledge of the encryption key. Interested reader can use [16] for a comprehensive introduction to cryptology.

To understand the advantages of an asymmetric cipher, let us imagine two persons– Alice and Bob, sending a secret message through the public mail.

With the asymmetric key system, Bob and Alice have separate padlocks. First, Alice asks Bob to send his open padlock to her through regular

mail, keeping his key to himself. When Alice receives the padlock, she uses it to lock her message, and sends the locked box to Bob. Bob can then unlock the box with his key and read the message from Alice. To reply, Bob must similarly get Alice's open padlock to lock the box before sending it back to her.

A commonly used variant of this is that Alice and Bob each owns two keys, one for encryption and one for decryption. The decryption key should not be deducible from the encryption key, so the encryption key can be published without compromising the security of encrypted messages. In the analogy above, Bob might publish on how to make a lock ("public key"), but the lock is impossible to be unlocked from these instructions about how to make a key. Those wishing to send messages to Bob can use the public key to encrypt the message and Bob can uses private key to decrypt it.

With a symmetric key system, Alice first puts the secret message in a box, and locks the box using a padlock to which she has a key. She then sends the box through regular mail to Bob. When Bob receives the box, he uses an identical copy of Alice's key to open the box and reads the message. Bob can then use the same padlock to send his secret reply.

Clearly, in an asymmetric key system Bob and Alice never need to send a copy of their keys to each other. This prevents a third party from copying a key while it is in transit. In addition, if Bob were careless and allows someone else to copy his key, Alice's messages to Bob would be compromised, but Alice's messages to other people would remain secret, since the other people would

be providing different padlocks for Alice to use.

In many practices, the asymmetric ciphers are used to securely exchange a **session key** for a symmetric cipher to be used for the actual communication. That is, the only plain text encrypted with the asymmetric cipher is the key for a symmetric cipher, and then the faster-running symmetric cipher is used for encryption of the actual message. The trick of using *session key* is very common. A *public*-key system is used to establish a shared key (the session key) for a (faster) *private*-key, through which the bulk of the communication will occur. After the message is sent, the session key is discarded and not reused. Thus the advantages of public-key ciphers can be realized while at the same time benefiting from the speed of symmetric ciphers.

Finally, note that all cryptosystems are subject to certain classes of attacks. For example, chosen-ciphertext attack exist for all public key cryptosystems, including RSA, ECC, and NTRUEnrypt. Such attacks do not affect the security of the underlying cryptographic algorithm, but instead they rely on sending specially constructed fake messages and observing the resulting decryptions. For each cryptosystem, there are straightforward techniques for defending against such attacks. The attacks are only effective if these simple precautions are not taken. In the following, we will show some attacks on the public key cryptosystems based on LLL algorithm, along with some recent advances in cryptosystems based on lattice reduction.

3.3.2 Knapsack and Subset Sum Problem

Knapsack ciphers use the same underlying mathematical problem, the *knapsack* or *subset sum problem.* The name comes from the maximization problem of choosing objects from a set of objects of various weights to fit into a bag (of maximum weight). One may look into the book *Kanpsack Problems: Algorithms and Computer Implementations* by Silvano Martello and Paolo Toth for more information about knapsack problems. In the following we will investigate the knapsack problem based cryptosystems and show how to attack such a system using the LLL algorithm.

A knapsack problem can be mathematically formulated as following:

Number the distinct objects from 1 to n. Let v_j be a measure of the value given by object j, a_j be its size and c the capacity of the knapsack, and introducing a vector of positive integer variables $x_j (j = 1, \ldots, n)$ such that $0 \leq x_j \leq b_j$, where b_j is the number of available jth object and subject to $b_j \leq c/a_j$, then our problem is equivalent to select the vector that maximizes the objective function

$$\sum_{j=1}^{n} v_j x_j$$

from all vectors x satisfying the constraint

$$\sum_{j=1}^{n} a_j x_j \leq c.$$

If b_j is some positive integer for each j, then we call it **Bounded**

Knapsack Problem, If $b_j = +\infty$ for each j, then we call it **Unbounded Knapsack Problem**. In particular, if $b_j = 1$, that is

$$x_j = \begin{cases} 1 & \text{if object } j \text{ is selected;} \\ 0 & \text{otherwise} \end{cases}, \tag{3.26}$$

then it is known as **0-1 Knapsack Problem**.

A particular case of the **0-1 Knapsack Problem** arises when $v_j = a_j$. The problem becomes to find a subset of weights whose sum is closest to, but not exceeding the capacity, i.e.

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} a_j x_j \\ \text{subject to} \quad & \sum_{j=1}^{n} a_j x_j \leq c, \end{aligned}$$

with x_js are defined as in (3.26). This is also called the *Subset Sum Problem.*

Both the general knapsack problem and the subset sum problem are NP-hard, this has led to attempts to use subset sum as the basis for public key cryptosystems, such as Merkle-Hellman cryptosystem. The Merkle-Hellman cryptosystem, invented by Ralph Merkle and Martin Hellman in 1978 [33], was one of the earliest public key cryptosystems. In 1982, Adi Shamir broke the basic Merkle-Hellman cryptosystem using Lenstra's linear programming algorithm, which was later improved in the LLL algorithm (see [31, 38]). Since then several attacks on more complicated knapsack cryptosystems have been

proposed. These attacks are all based on the idea of recovering the trapdoor information concealed in the sizes $\{a_i : 1 \leq i \leq n\}$. Adleman extended Shamir's work by treating the cryptographic problem as a lattice problem rather than a linear programming problem.[1]. In the following, we use the Merkle-Hellman cryptosystem to show how to break a simple version of Merkel-Hellman cryptosystem (singly-iterated Merkle-Hellman cryptosystem) using the LLL algorithm.

3.3.3 Merkle-Hellman Cryptosystem and An Attack Using LLL Algorithm

Let us now define a slightly different version of subset sum problem from above, but is indeed used in cryptography:

Definition 4 (Subset Sum Problem) *Given a vector* $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ *consisting all positive integers, and* s *a target positive integer, determine if there is vector* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ *with* $x_1, x_2 \ldots, x_n \in \{0, 1\}$ *satisfying*

$$x_1 a_1 + x_2 a_2 + \cdots + x_n a_n = s.$$

Since the subset sum problem is NP-complete, so it is supposed to be hard to determine if there the vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as defined in the definition exists. The trivial solution is to try all 2^n possible values for $\curvearrowleft = (x_1, x_2, \ldots, x_n)$. A better method is to compute the common element

$$S_1 = \left\{ \sum_{j=1}^{\lfloor n/2 \rfloor} x_j a_j \ : \ x_j = 0 \ or \ 1 \ \forall j \right\}$$

and

$$S_2 = \left\{ s - \sum_{j>\lfloor n/2 \rfloor}^{n} x_j a_j \ : \ x_j = 0 \ or \ 1 \ \forall j \right\},$$

this procedure takes $O(n2^{n/2})$ operations and is still the known fastest algorithm for the general knapsack problem. A natural way to build up a knapsack cryptosystem is to use the vector $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ as public key and $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ as the plain text, so the ciphertext is

$$s = \sum_{j=1}^{n} x_j a_j.$$

But then not only the unauthorized receiver has to solve the knapsack problem if he is trying to decipher the message, the intended receiver is also faced to solving the knapsack problem. To bypass this problem, Merkle and Hellman proposed to use a superincreasing sequence for vector $\mathbf{a}$ and conceal it by some sort of invertible transformation.

If

$$a_j > \sum_{i=1}^{j-1} a_i \quad for \ 2 \leq j \leq n,$$

then a_js form a superincreasing sequence. Moreover, we can easily find x_n, since

$$x_n = 1 \quad if\ and\ only\ if \quad s > \sum_{j=1}^{n-1} a_j.$$

Once we find x_n, we have a reduced knapsack problem

$$s - x_n a_n = \sum_{j=1}^{n-1} x_j a_j, \quad x_j \in \{0,1\}, \quad 1 \leq j \leq n-1,$$

thus we can recursively retrieve the entire message $(x_1, x_2, \ldots, x_n)$. But if we directly use $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ as public key, then an attacker can decipher the message as easily as the intended receiver. So we need a one-way trapdoor to conceal this information.

In basic Merkle-Hellman cryptosystem, Bob choose a superincreasing sequence $b_1, b_2, \ldots, b_n$ with $b_1 \approx 2^n$ and $b_n \approx 2^{2n}$, and $M, W \in \mathbb{Z}$ with $M > b_1 + b_2 + \cdots + b_n$ and $gcd(M, W) = 1$, and a permutation π of the integers $\{1, \ldots, n\}$. Then Bob's public key is $\{a_1, a_2, \ldots, a_n\}$ with

$$a_j = W b_{\pi(j)} (mod M),$$

while his private key is b_j, M, W and the permutation π.

Now if Alice's plain text is $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}$, then her ciphertext will be

$$s = \sum_{j=1}^{n} x_j a_j.$$

To decrypt, Bob then computes

$$\begin{aligned} c &= sW^{-1}(modM) \\ &= \sum_{j=1}^{n} x_j a_j W^{-1}(modM) \\ &= \sum_{j=1}^{n} x_j (W b_{\pi(j)}) W^{-1}(modM) \\ &= \sum_{j=1}^{n} x_j b_{\pi(j)}(modM). \end{aligned}$$

Since $M > \sum b_j$, so c exactly equals the sum, i.e. $c = \sum_{j=1}^{n} x_j b_{\pi(j)}$. Also $b_1, b_2, \ldots, b_n$ is superincreasing, so Bob can easily solve this knapsack problem and recover the plain text x.

The choice of $b_1 \approx 2^n$ and $b_n \approx 2^{2n}$ is supposed to ensure that an attacker has a great number of possibilities for each parameter, and hence anyone who does not know M, W and b_j has great difficulty in solving x even the general method used for generating the trapdoor knapsack vector **a** is known by the public. Notice that if b_js are too large, then the knapsack system will not efficient as the M has to be large but n bits information will be encoded into roughly $log_2 M$ bits. Interested readers can find more information and examples in [33].

In the following, we will show how to convert a knapsack problem to a lattice problem, and how to break the basic Merkle-Hellman cryptosystem using one of the improvement from that of Adleman's [1] due to Lagarias and

Odlyzko in [30].

Consider a knapsack problem to be solved:

$$s = x_1 a_1 + x_2 a_2 + \cdots + x_n a_n,$$

we can construct the lattice generated by the matrix[5]

$$V = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & 1 & 0 \\ a_1 & a_2 & a_3 & \dots & a_n & -s \end{bmatrix}.$$

Clearly, the columns of the matrix are linearly independent and form a basis of the lattice, say L, of $\mathbb{Z}^{n+1}$. If $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0,1\}^n$ solves s, let $v_1, v_2, \dots, v_{n+1}$ represents the columns in the matrix, then

$$\sum_{j=1}^{n} x_j v_j + v_{n+1} = (x_1, x_2, \dots, x_n, 0)^T. \tag{3.27}$$

Since the x_js are 0 or 1, this vector is very short. If it is indeed the shortest one in L, then LLL or one of its variants may be able to find it. Fortunately, the a_js are large, so one could expect that most vectors in the

[5] We slightly changed the matrix form to fit our needs.

lattice would be large, and the vector 3.27 might be the shortest one. In [30], it was shown that if a_js are chosen at random with

$$a_j \approx 2^{\beta n}, \quad 1 \leq j \leq n$$

where β is any constant > 1.54725, then the vector 3.27 is the shortest non-zero vector with high probability. This bound was later improved to be $\beta > 1.063$ in [10].

Because the LLL algorithm is only guaranteed to return an approximated shortest vector, it can not be proved to break all the instances with good lattices. However, the LLL algorithm performs better in practice than in theoretical bound, so it might be prudent to say that we could solve most low-density knapsack based cryptosystems. To help make these ideas clearer, we will give two small but concrete examples in the following (without considering the permutation).

Example 1

Let $n = 4$, the private key $\mathbf{b} = (3, 11, 19, 35)$;

choosing $M = 3001$;

taking $W = 2007$, therefore $W^{-1} = 474$.

So the public key is:

$\mathbf{a} = W\mathbf{b}(mod\ M) = (19, 1070, 2121, 1222)$.

If Bob is going to send Alice the plain text 1010, then the ciphertext Alice will be receiving is:

$$s = 19 * 1 + 1070 * 0 + 2121 * 1 + 1222 * 0 = 2140.$$

Alice then uses s to find the plain text by our algorithm described in 1.2.2 and computes

$$\begin{aligned} c &= sW^{-1}(modM) \\ &= 2140 * 474(mod3001) \\ &= 22, \end{aligned}$$

and uses c to find the plain text by

$$\begin{aligned} 22 - 35(0) = 22 \quad & x_4 = 0 \\ 22 - 19(1) = 3 \quad & x_3 = 1 \\ 3 - 11(0) = 3 \quad & x_2 = 0 \\ 3 - 3(1) = 0 \quad & x_1 = 1. \end{aligned}$$

While an unauthorized user knows:

the public key $\mathbf{a} = (19, 1070, 2121, 1222)$;

received ciphertext 2140.

He or she wants to find $\mathbf{x} = (x_1, \ldots, x_4) \in \{0, 1\}$ such that

$$19x_1 + 1070x_2 + 2121x_3 + 1222x_4 = 2140.$$

Having written the matrix

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 19 & 1070 & 2121 & 1222 & -2140 \end{bmatrix},$$

and applied the improved LLL algorithm we described, he or she obtains a reduced matrix $\hat{V}$

$$\hat{V} = \begin{bmatrix} 1 & 0 & -2 & -3 & 7 \\ 0 & -2 & 0 & 1 & 0 \\ 1 & 0 & 2 & 4 & -8 \\ 0 & 0 & -7 & 1 & -2 \\ 0 & 0 & 0 & 9 & 1 \end{bmatrix}.$$

Clearly, the 1st column in red of the matrix is the only solution in correct form and is indeed the right answer, thus the message is deciphered.

Example 2 ([3])

Let $n = 9$, the private key $\mathbf{b} = (2, 5, 9, 21, 45, 103, 215, 450, 946)$;

choosing $M = 2003$;

taking $W = 1289$, therefore $W^{-1} = 317$.

So the public key is:

$\mathbf{a} = W\mathbf{b}(mod\ M) = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570)$.

If Bob is going to send Alice the plain text 101100111, then the cipher-text Alice will be receiving is:

$s = 575*1+436*0+1586*1+1030*1+1921*0+569*0+721*1+1183*1+1570*1 = 6665.$

Alice then uses s to find the plain text by our algorithm described in 1.2.2 and computes

$$\begin{aligned} c &= sW^{-1}(modM) \\ &= 6665 * 317(mod2003) \\ &= 1643, \end{aligned}$$

and uses c to find the plain text by

$$\begin{aligned} 1643 - 946(1) = 697 \quad & x_9 = 1 \\ 697 - 450(1) = 247 \quad & x_8 = 1 \\ 247 - 215(1) = 32 \quad & x_7 = 1 \\ 32 - 103(0) = 32 \quad & x_6 = 032 - 45(0) = 32 \qquad x_5 = 0 \end{aligned}$$

$$\begin{aligned} 32 - 21(1) &= 11 \quad x_4 = 1 \\ 11 - 9(1) &= 2 \quad x_3 = 1 \\ 2 - 5(0) &= 2 \quad x_2 = 0 \\ 2 - 2(1) &= 0 \quad x_1 = 1. \end{aligned}$$

While an unauthorized user knows:

the public key $\mathbf{a} = (575, 436, 1586, 1030, 1921, 569, 721, 1183, 1570)$;

received ciphertext 6665.

He or she wants to find $\mathbf{x} = (x_1, \ldots, x_9) \in \{0, 1\}$ such that

$$575x_1+436x_2+1586x_3+1030x_4+1921x_5+569x_6+721x_7+1183x_8+1570x_9 = 6665.$$

If he or she writes the matrix

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 575 & 136 & 1586 & 1030 & 1921 & 569 & 721 & 1183 & 1570 & -6665 \end{bmatrix},$$

after applying the improved LLL algorithm, he or she obtains a reduced matrix $\hat{V}$

$$\hat{V} = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 & -1 & -1 & 0 & 1 & -1 \\ 0 & -1 & 0 & 1 & -1 & -1 & -1 & 1 & 0 & 1 \\ 0 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & 0 & -1 & -1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ -1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & -2 \\ 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & 1 & 2 \\ 1 & 0 & -1 & -1 & 0 & -1 & -2 & 0 & 0 & 0 \end{bmatrix},$$

Clearly, the 7th column in red of the matrix is the only solution in correct form and is indeed the right answer, thus the message is deciphered.

The examples given were extremely small in size and were only intended to illustrate the ideas. It was thought that $n = 100$ is the bottom end of the usable range for secure systems in [33] though in fact almost all cryptosystems based on Knapsack problem have been broken so far. Interested reader can see the comparisons among the variants of the LLL algorithm in deciphering the Merkle-Hellman method encrypted messages.

Chapter 4

An LLL Reduction Aided Sphere Decoding

The idea of formulating the detection of a lattice type modulation transmitted over a linear channel as the so-called universal lattice decoding problem is particular attractive for bandwidth efficient modulations, such as L-PAM and L-QAM, due to its various desirable properties [35]. In this chapter, we show how to solve the ILS problem using the LLL reduction aided sphere decoding algorithm and simulate the new algorithm to show that it achieves a better performance than the maximum likelihood sphere decoding.

4.1 LLL Reduction Aided Sphere Decoding

Recall the integer least-squares problem is defined as

$$\min_{\mathbf{s}\in\mathcal{D}\subset\mathbb{Z}^m} \|\mathbf{x} - H\mathbf{s}\|_2,$$

where $\mathbf{x} \in \mathbb{R}^n$, $H \in \mathbb{R}^{n\times m}$, and $\mathbb{Z}^m$ denotes the m-dimensional integer lattice and $\mathcal{D} \subset \mathbb{Z}^m$.

Since the noiseless received signal $H\mathbf{s}$ can be considered as a point in the lattice space generated by the lattice generating matrix H if the entries of $\mathbf{s}$ are taken from the set of integers, thus only a finite subset of this lattice is actually used as practical systems typically have only limited average and peak output power. As we proved at the beginning of the chapter 3, for any lattice, the bases are not unique, it is always possible to find a unimodular matrix M, such that if H is a basis then $\widehat{H} = HM$ is also a basis. Specifically, a point in the basis H represented by $\mathbf{s}$ can be represented in $\widehat{H}$ by $\mathbf{z} = M^{-1}\mathbf{s}$. Effectively, we have transformed our ILS in (2.2) into the following:

$$\min_{\mathbf{s}\in\mathcal{D}\subset\mathbb{Z}^m} \|\mathbf{x} - \widehat{H}M^{-1}\mathbf{s}\|_2 \implies \min_{\mathbf{s}\in\mathcal{D}\subset\mathbb{Z}^m} \|\mathbf{x} - \widehat{H}\mathbf{z}\|_2.$$

With this basis change, the receiver first compensates the new channel $\widehat{H} = HM$, then $\mathbf{s}$ is produced by $M^{-1}\mathbf{z}$.

Now, we can re-address our sphere decoding algorithm as the followings.

Consider the QR factorization of $\widehat{H}$:

$$\widehat{H} = \widehat{Q} \begin{bmatrix} \widehat{R} \\ 0_{(n-m)\times m} \end{bmatrix},$$

where $\widehat{R}$ is an $m \times m$ upper triangular matrix and $\widehat{Q} = [\widehat{Q}_1 \quad \widehat{Q}_2]$ is an $n \times n$ orthogonal matrix, in which the matrices $\widehat{Q}_1$, $\widehat{Q}_2$ represent the first m and last $n - m$ orthonormal columns of $\widehat{Q}$ respectively.

The condition (2.3) then can be written as

$$\begin{aligned} \rho^2 &\geq \left\| \mathbf{x} - [\widehat{Q}_1 \quad \widehat{Q}_2] \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix} \mathbf{z} \right\|^2 \\ &= \left\| \begin{bmatrix} \widehat{Q}_1^T \\ \widehat{Q}_2^T \end{bmatrix} \mathbf{x} - \begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix} \mathbf{z} \right\|^2 \\ &= \|\widehat{Q}_1^T \mathbf{x} - \widehat{R}\mathbf{z}\|^2 + \|\widehat{Q}_2^T \mathbf{x}\|^2. \end{aligned} \tag{4.1}$$

Note that for any QR factorization of H, we have

$$\begin{aligned} \|Q^T \mathbf{x}\|^2 &= (Q^T \mathbf{x})^T (Q^T \mathbf{x}) \\ &= \mathbf{x}^T Q Q^T \mathbf{x} \\ &= \mathbf{x}^T \mathbf{x} \\ &= \|\mathbf{x}\|^2. \end{aligned}$$

That is,

$$\begin{aligned}\|\mathbf{x}\|^2 &= \left\| \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} \mathbf{x} \right\|^2 \\ &= \left\| \begin{bmatrix} Q_1^T\mathbf{x} \\ Q_2^T\mathbf{x} \end{bmatrix} \right\|^2 \\ &= \|Q_1^T\mathbf{x}\|^2 + \|Q_2^T\mathbf{x}\|^2.\end{aligned}$$

so the above inequality (4.1) becomes

$$\begin{aligned}\rho^2 - \|\widehat{Q}_2^T\mathbf{x}\|^2 &= \rho^2 - \|\mathbf{x}\|^2 + \|\widehat{Q}_1^T\mathbf{x}\|^2 \\ &\geq \|\widehat{Q}_1^T\mathbf{x} - \widehat{R}\mathbf{z}\|^2.\end{aligned}$$

Thus, we only need the compact QR decomposition for $\widehat{H}$: $\widehat{H} = \widehat{Q}_1\widehat{R}$. Defining $y = \widehat{Q}_1^T\mathbf{x}$ and $\widehat{\rho}^2 = \rho^2 - \|\mathbf{x}\|^2 + \|\widehat{Q}_1^T\mathbf{x}\|^2$, we have:

$$\widehat{\rho}^2 \geq \sum_{i=1}^{m} \left(y_i - \sum_{j=i}^{m} \widehat{r}_{ij} z_j \right)^2 . \tag{4.2}$$

Recall that after the LLL reduction, the entries on the diagonal of the upper triangular matrix $\widehat{R}$ are all positive real numbers, so we can define our new LLL reduction aided sphere decoding algorithm from (4.2) as:

Input: $\widehat{Q}_1$, $\widehat{R}$, $\mathbf{z}$, $\mathbf{y} = Q_1^T\mathbf{x}$, $\rho = \max(\|\widehat{h}_1\|_2, \|\widehat{h}_2\|_2, \ldots, \|\widehat{h}_m\|_2)$.

1. Set $k = m$, $\widehat{\rho}_k^2 = \rho^2 - \|\mathbf{x}\|^2 + \|\widehat{Q}_1^T\mathbf{x}\|^2$, $y_{k|k+1} = y_k$

2. (Bounds for z_k) Set $UB(z_k) = \lfloor \frac{\widehat{\rho}_k + y_{k|k+1}}{\widehat{r}_{k,k}} \rfloor$, $z_k = \lceil \frac{-\widehat{\rho}_k + y_{k|k+1}}{\widehat{r}_{k,k}} \rceil - 1$.

3. (Increase z_k) $z_k = z_k + 1$, If $z_k \leq UB(z_k)$, go to 5, else go to step 4.

4. (Increase k) $k = k + 1$, if $k > m$, return results and terminate, else go to 3.

5. If $k = 1$, go to next step. Else, save z_k, and set $k = k - 1$, $y_{k|k+1} = y_k - \sum_{j=k+1}^{m} \widehat{r}_{kj} z_j$, $\widehat{\rho}_k^2 = \widehat{\rho}_{k+1}^2 - \left(y_{k+1|k+2} - \widehat{r}_{k+1,k+1} z_{k+1}\right)^2$ and go to 2.

6. Solution found. Save z_k and go to 3.

Once the optimal solution $\mathbf{z}$(s) are found, then we can simply use $M^{-1}\mathbf{z}$ to get back the transmitted signal $\mathbf{s}$.

4.2 Experiment Setup

In order to perform some numerical simulations to compare the new LLL reduction aided sphere decoding algorithm with the maximum likelihood sphere decoding algorithm, we specify the data sources and some terminologies in this section with some of the contents in this section can be found in [26, 32].

For simplicity, we assume the transmitted signals are the symbols from L-PAM (Pulse-amplitude modulation), but the algorithm can be readily applied to complex L-QAM. Here, PAM refers a form of signal modulation where the message information is encoded in the amplitude of a series of signal pulses. For example, A two bit modulator (4-PAM) will take two bits at a time and

map the signal amplitude to one of four possible levels, for example, -3 volts, -1 volt, 1 volt, and 3 volts. Pulse-amplitude modulation is widely used in baseband transmission of digital data, the widely popular Ethernet communication standard is a good example of PAM usage. In particular, we choose 4-PAM constellation with the SNR (Signal to Noise Ratio) at $0db$, $5db$, $10db$, $15db$, $20db$, 25 to compare the two algorithms' symbol error rate (SER) and efficiency.

Furthermore, the entries of the noise vector $\mathbf{v}$ are independently and identically distributed white Gaussian random variables with zero mean and unit variance and is applied on the receiving transmitters. The channel coefficients are generated according to Gaussian distribution with zero mean and unit variance and the channel matrix is of Toeplitz form in (1.8). As with the maximum likelihood method, we always set the probability Pr in (2.10) as 0.95 and look up the probability table for the Chi-square distribution in [6] or [27] to find the value of ρ^2/δ^2, where ρ is the correspondent searching radius and $\delta = 1$ by the assumption that the noise has a unit variance. [35]

Finally, we compare the algorithms' efficiency and accuracy in terms of running time and symbol error rate (SER) respectively. The transmitted symbols satisfy the power constrain function $P_{ower} \sum_{i=1}^{n} S_i^2 P_i = 1$ and thus the $P_{ower} = 1/\sqrt{5}$ and $P_{ower} = 1/\sqrt{21}$ for 4-PAM and 8-PAM respectively to observe the fact that as the antennas in a constellation increases the power of the signal decreases. Effectively, the mathematical model for the ILS has

become

$$10^{\frac{SNR}{10}} P_{ower} H\mathbf{s} + \mathbf{v} = \mathbf{x},$$

where H is the channel matrix, $\mathbf{v}$ is the additive noise vector, $\mathbf{s}$ is the transmitted signal and $\mathbf{x}$ is the received signal, SNR and P_{ower} are defined as in above.

4.2.1 Simulation Results and Conclusions

In this section, we listed all the test results to compare the two algorithms. The programs presented in the *Appendix* were written in *MATLAB v.7.1* (student edition) on a Sun-microsystem computer. In the programs, we implicitly used the fact that the channel matrix is of the form Toeplitz, which is often well conditioned and forms a basis for the generated space. The former minimizes the introduced error propagation due to the calculation of the QR factorization and the latter enable us to implement the LLL reduction algorithm on the channel matrix.

The following graphs show the comparison of the symbol error rate(SER) and the time consumed for MLSD and LRSD respectively. For statistical sufficiency purpose, we run each algorithm $10,000$ time and take the average SER and time consumed for each algorithm. Notice that the simulation results may vary at each run, but the general trend reflected in the following graph has been preserved at all times.

It is easy to see that the SER is decreasing for both algorithms when

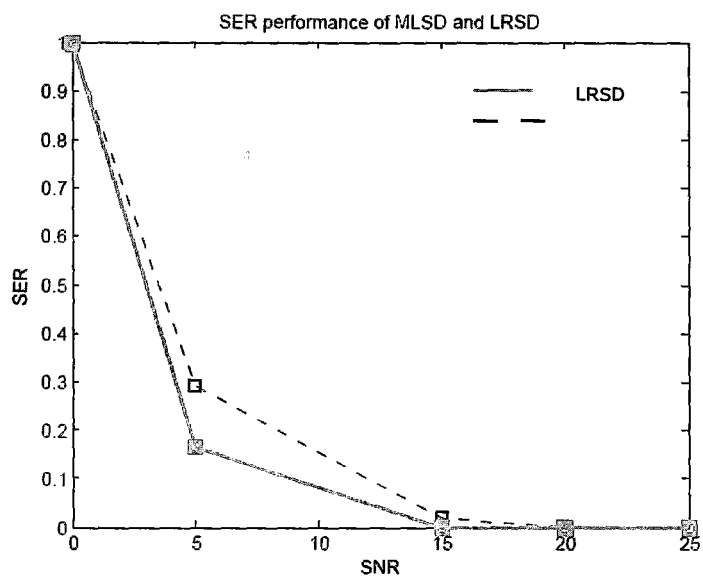

Figure 4.1: Error Rate Comparison of the MLSD and LRSD with 4-PAM

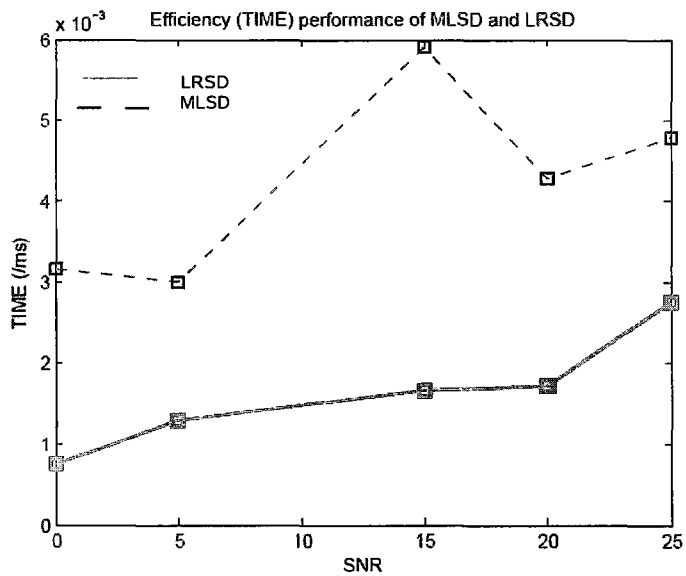

Figure 4.2: Efficiency Comparison of the MLSD and LRSD with 4-PAM

the SNR increases as it is well known that SER is a decreasing function of SNR.

The increasing of SNR is equivalent to adding power on the transmitted signal, so the relative noise impact on the $10^{\frac{SNR}{10}} P_{ower} H\mathbf{s}$ is smaller. Thus, for the MLSD, even with a smaller probability (therefore a smaller searching radius and fewer number of lattice points to enumerate) we can find a nearby lattice point to the skewed one. Similarly, having a relative orthogonal and shortened searching base, the LRSD method has a comparable SER with MLSD algorithm and rendered the fact that SER is a decreasing function of SNR.

Furthermore, for the LRSD, basis vectors do not change as it only depends on the channel matrix which is not changing by our assumption. When the signal to noise ratio at a high level, there is a relative large SER at first, but as the signal to noise ratio increases, the search radius does not change, thus yields a relative smaller searching radius comparing to the MLSD methods, which explains why the LRSD has a better performance both in terms of efficiency and accuracy than the MLSD as SNR increases. Recall that any skewed lattice point has to sit in between two lattice point in the reduced lattice space, so the LLL reduction aided sphere decoding must yield a closest lattice point within the norm of the longest reduced lattice vector, which is unlike the maximum likelihood that an inappropriate probability choice we may not have an (optimal) solution at all.

It must be noted that although the LRSD achieves comparable SER

performance with a increased efficiency and accuracy than the MLSD, the LRSD's performance highly depends on the basis reduction process while this process is currently developed without considering that the floating point arithmetics. To this end, we also would like to point out that the optimal solution is chosen by calculating the minimum norm of $\|Hs - H\widehat{s}\|_2$, which in MLSD is decided by the probability and in LRSD is decided by the largest reduced lattice vector. This choice is not necessary true in practice, but can be easily adapted when considering other choosing criteria.

4.2.2 Further Work

Keep in mind that the lattice point enumeration is the basis of the best known algorithm to solve exactly the shortest vector problem and also known in MIMO as Sphere Decoding algorithm. Though the lattice-reduction-aided detectors have been proposed for multiple-input multiple-output (MIMO) communication systems to give performance with full diversity like maximum likelihood optimal receiver but with complexity similar to linear receiver, these proposals are based on the original LLL reduction algorithm that was intended for real lattice basis even though the channel matrices are inherently complex-valued. Besides, the original LLL reduction was not developed in consideration of the floating point arithmetics as we pointed earlier, so it is inevitable that the round off errors would be large in high dimensional lattice. Thus, a more stable LLL reduction or basis reduction algorithm can at least improve the

accuracy of the searching radius and reduce the computations needed in the sphere decoding.

Also notice that the complexity of the sphere decoding can be improved by further exploiting the Toeplitz structure of the channel matrix. Remember that in order to use the sphere decoding, all we have to do is to perform the QR factorization to obtain the upper triangular matrix R, which can be sped up using the Toeplitz structure in both the MLSD and the LRSD.

Bibliography

[1] L. M. Adleman. On breaking generalized knapsack public key cryptosystems. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 402–412, 1983.

[2] K. S. Arun. A unitary constrained total least squares problem in signal processing. In *SIAM Journal: Matrix Analysis*, volume 13(3), pages 729–745, 1989.

[3] J. Bakker. A simplified version of lenstra's integer programming algorithm and some applications. Math184 Couse Notes,Computer Science Department, University of California at San Deigo, Spring 2004.

[4] Inaki Berenguer, Jaime Adeane, Ian Wassell, and Xiaodong Wang. Lattice reduction aided receivers for mimo-ofdm in spatial multiplexing systems. In *Personal, Indoor and Mobile Radio Communications(PIMRC 2004), 15th IEEE International Symposium*, volume 2, pages 1517–1521, Sep. 2004.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. In *Communications ICC 93, IEEE International Conference*, volume 2, pages 1064 – 1070, 1993.

[6] William H. Beyer. *Handbook of tables for probability and statistics.* The Chemical Rubber Co., 2nd edition, 1968.

[7] E. Biglieri, Giprgio Taricco, and Antonia Tulino. Performance of space-time codes for a large number of antennas. In *IEEE Transactions on Information Theory*, volume 48, No.7, pages 1794–1803, July 2002.

[8] J.W.S. Cassels. *An Introduction to the Geometry of Numbers.* Classics in Mathematics. Springer-Verlag, 1st edition, 1997.

[9] Henri Cohen. *A course in Computational Algebraic Number Theory.* Graduate Texts in Mathematics. Springer-Verlag, 1993.

[10] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithm. In *EUROCRYPT 1991*, 1991.

[11] James W. Demmel. *Applied Numerical Linear Algebra.* SIAM, 1997.

[12] W. Diffie and Martin E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, pages 644–654, 1976.

[13] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. In *Mathematics of Computation*, volume 44, No.170, pages 463–471, April 1985.

[14] Eli Fogel. Total least squares for toeplitz structures. In *21st IEEE Conference on Decision and Control*, volume 21, pages 1003–1004, 1982.

[15] G. J. Foschini and M. J. Gans. On limits of wireless communications in a fading enviroment when using multiple antennas. In *Wireless Personal Communications*, volume 6(3), page 311, 1998.

[16] Paul Garrett. *Making, Breaking Codes: an introduction to cryptology.* Prentice Hall, 2001.

[17] L. J. Gleser. Estimation in a multivariate "error in variables" regression model: large sample results. In *The Annals of Statistics*, volume 9, pages 24–44, 1981.

[18] G. H. Golub and C. F. Van Loan. An analysis of the total least squares problem. In *SIAM Journal: Matrix Analysis*, volume 17(6), pages 883–893, 1980.

[19] G. H. Golub and C. F. Van Loan. *Matrix Computation.* The Johns Hopkins University Press, 3rd edition, 1996.

[20] B. Hassibi and H. Vikalo. Expected complexity of the sphere decoding algorithm. In *IEEE Trans. on Signal Processing*, 2001.

[21] B. Hassibi and H. Vikalo. On the sphere decoding algorithm: Part i, the expected complexity. In *IEEE Transactions on Signal Processing*, volume 53, No.8, pages 2806–2818, Aug. 2005.

[22] Micheal T. Heath. *Scientific Computing: an introductory survey.* McGraw-Hill higher education, 2nd edition, 2002.

[23] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem: computational aspects and analysis.* Frontiers in Applied Mathematics. SIAM.

[24] Ravi Kannan. Algorithmic geometry of numbers. Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

[25] Ravi Kannan. Improved algorithm for integer programming and related lattice problems. In *Annual ACM Symposium on Theory of Computing*, pages 193 – 206, 1987.

[26] Steven M. Kay. Fundamentals of statistical signal processing. In *Detection Theory*, volume 2. Prentice Hall, 1998.

[27] Stephen Kokoska and Christopher Nevison. *Statistical Tables and Formulae.* Springer-Verlag, 1988.

[28] Henrik Koy and Claus P. Schnorr. Segment lll-reduction with floating point orthogonalization. In *Lecture Notes In Computer Science*, volume

2146, pages 81–96. International Conference on Cryptography and Lattices, 2001.

[29] Volker Kuhn. *Wireless Communications over MIMO channels.* Signal Processing. Prentice Hall, 1st edition, 2006.

[30] J. C. Lagarias and A. M. Odlyzko. Solving low-density subset sum problems. In *In Proceedings of the 24th IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1983.

[31] A. K. Lenstra, H.W. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 1982.

[32] Richard G. Lyons. *Understanding Digital Signal Processing.* Prentice Hall, 2nd edition, 2004.

[33] Ralph. C. Merkle and Martin E. Hellman. Hiding information and signatures in trapdoor knapsacks. In *IEEE Transactions on Information Theory*, volume 24, pages 525–530, 1978.

[34] Amin Mobasher, Mahmoud Taherzadeh, Renata Sotirov, and Amir K. Khandani. A near maximum likelihood decoding algorithm for mimo systems based on semi-definite programming.
citeseer.ist.psu.edu/mobasher05near.html.

[35] Wai Ho Mow. Universal lattice decoding: principle and recent advances. In *Wireless Communications and Mobile Computing*, 2003.

[36] Phong Nguyen and Damien Stehle. Segment lll-reduction with floating point orthogonalization. In *Lecture Notes In Computer Science*, volume 3494, pages 215–233. Springer-Verlag, 2005.

[37] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing.* Signal Processing. Prentice Hall, 2nd edition, 2006.

[38] A. Paz. A simplified version of h. w. lenstra's integer programming algorithm and some applications.
http://techreports.lib.berkeley.edu/accessPages/CSD-83-116.html.

[39] Theodore S. Rappaport. *Wireless Communications.* Prentice Hall, 1st edition, 1996.

[40] Claus P. Schnorr. A more efficient algorithm for lattice basis reduction. In *Journal of Algorithms*, pages 47–62, 1988.

[41] Claus P. Schnorr. New practical algorithms for the approximate shortest lattice vector. Technical report, Fachbereiche Mathematik/informatik, Frankfurt University, PSF 41932, Germany, 2001.

[42] Claus P. Schnorr. Fast lll-type lattice reduction. In *Information and Computation*, volume 204, 2006.

[43] Claus P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Fachbereiche Mathematik/informatik, Frankfurt University, Postfach 111932, Germany, 1993.

[44] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley-interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1998.

[45] Gilbert Strang. *Linear Algebra and Its Applications.* Brooks Cole, 3rd edition, 1988.

[46] Haris Vikalo and Babak Hassibi. Maximum-likelihood sequence dectection of multiple antenna systems over dispersive channels via sphere decoding. In *EURASIP Journal on Applied Signal Processing*, pages 525–531, 2002.

[47] E. Viterbo and J. Boutros. A universal lattice code decoder for fading channels. In *IEEE Trans. on Information Theory*, volume 45, pages 1639 – 1642, 1997.

[48] Christoph Windpassinger and Robert Fishcer. Low complexity near maximum likelihood detection and precoding for mimo system using lattice reduction. In *Proceedings of IEEE Information Theory Workshop*, March 2003. Paris, France.

[49] Huan Yao and Gregory W. Wornell. Lattice-reduction-aided detectors for mimo communication systems. In *Global Telecommunications Conference*, volume 1, pages 424–428, Nov. 2002.

Appendix: Prgrams Used for the Simulations

The LLL Reduction Algorithm Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo    April. 11, 2008
%McMaster University    Department of computing and software
%
%Purpose: Useing LLL algorithm to reduce the given lattice space
%
%Input: basis B to be reduced
%
%Output: B: an LLL reduced basis that overwrite the original B
%        Q: orthogonal basis
%        R: upper triangular basis
%        U: a unimodular matrix, such that B*U = QR, where this B is the
%        original B to be reduced
%
%Precodition: input basis B must be of a basis or the program is
%unpredicatable.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [B, Q, R, U]= LLLReduction(B)

%initiate the current position k and maximum visited position kmax and
```

```
%mu(s). record the scalar product of two vectors as in S(i), for 1<= i <=n.
k = 2; kmax = 1; b1 = B(:, 1);

[m, n] = size(B); mu = eye(n, n); S = zeros(1, n); S(1) = sum(b1.*b1);

BS =  zeros(m, n); BS(:, 1) = b1; U = eye(n, n);

%step 4 in the algorithm: walking through all the column vectors.
while k <= n

    %step 2 in the algorithm: check if we can work forward.
    if k > kmax
        kmax = k;
        bk = B(:, k);
        bsk = bk;
        for j = 1: k-1
            mu(k, j) = sum(bk .*BS(:, j))/S(j);
            bsk = bsk - mu(k, j)* BS(:, j);
        end
        BS(:, k) = bsk;
        S(k) = sum(bsk .*bsk);
    end
```

```
%step 3 in the algorithm: first execute reduction.
if (abs(mu(k, k-1))-0.5) > 0
    r = round(mu(k, k-1));
    B(:, k) = B(:, k) - r * B(:, k-1);
    mu(k, k-1) = mu(k, k-1) - r;
    U(:, k) = U(:, k) - r * U(:, k-1);
    if k>2
        for j = 1: k-2
            mu(k, j) = mu(k, j) - r * mu(k-1, j);
        end
    end
end

%step3 in the algorithm: next check if we need do swap.
if (0.75 - mu(k, k-1) * mu(k, k-1)) * S(k-1) - S(k) > 0
    %if needed, then do the pairwise swap for b_k and b_{k-1}.
    temp1 = B(:, k);
    B(:, k) = B(:, k-1);
    B(:, k-1) = temp1;

    %and the UK and Uk-1
    utemp = U(:, k-1);
```

```
U(:, k-1) = U(:, k);
U(:, k) = utemp;

temp = mu(k, k-1);

if k>2
    for j = 1: k-2
        %exchange mu(k, j) with mu(k-1, j)
        temp2 = mu(k, j);
        mu(k, j) = mu(k-1, j);
        mu(k-1, j) = temp2;
    end

    %change the orthognal basis k-1 first:
    vp = BS(:, k-1); %c_{k-1}^* = b_k^* + mu(k, k-1) b_{k-1}^*
    BS(:, k-1) = BS(:, k) + mu(k, k-1) * vp;

    tempS = S(k) + temp * temp * S(k-1);
    mu(k, k-1) = temp * S(k-1) / tempS;
    S(k) = S(k-1) * S(k)/tempS;
    S(k-1) = tempS;

    %then the kth orthognal vector:
```

```
        BS(:, k) = vp - mu(k, k-1) * BS(:, k-1);
    else
        BS(:, k-1) = temp1;
        S(k-1) = sum(temp1 .*temp1);
        temp1 = B(:, k);
        mu(k, k-1) = sum(temp1 .*BS(:, k-1))/S(k-1);
        bsk = temp1 - mu(k, k-1)*BS(:, k-1);
        BS(:, k) = bsk;
        S(k) = sum(bsk .*bsk);
    end


    if k < kmax
        for i = k+1: kmax
            t = mu(i, k);
            mu(i, k) = mu(i, k-1) - temp*t;
            mu(i, k-1) = t + mu(k, k-1) * mu(i, k);
        end
    end

    k = max(2, k-1);
else
    if k > 2
```

```
            for d = k-2: -1: 1
                if abs(mu(k, d)) - 0.5 > 0
                    r = round(mu(k, d));
                    B(:, k) = B(:, k) - r * B(:, d);
                    mu(k, d) = mu(k, d) - r;
                    U(:, k) = U(:, k) - r * U(:, d);
                    if d >1
                        for q = 1: d-1
                            mu(k, q) = mu(k, q) - r * mu(d, q);
                        end
                    end
                end
            end
        end
        k = k+1;
    end
end

Q = BS; R = mu';
%finally, it is easy to get the QR decomposition of the reduced base:
for i=1 : n
    intermediate = sqrt(S(i));
    Q(:, i) = Q(:, i)/intermediate;
```

```
    R(i, :) = R(i, :)*intermediate;
end
%mu
%det(U)
```

The MLSD Algorithm Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo April 11 2008
%
%McMaster University, Department of Computing and Software
%
%Purpose: implement Maximum likelihood sphere decoding algorithm
%
%Inputs:
%        dets: received signal vecto with all entries are real
%              the vector must be a column vector
%        r: initial searching radius for the sublattice
%        H: channel matrix
%
%Output:
%        S: sub-lattice space with all potential solutions
%        time: the time used  to run the SD algorithm
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [S, time] = mlsd(dets, H, r)
```

```
%initial the beginning time:
time = cputime;

%S is a matrix memorizing the s_k(s) at each step of k, from s_m
%to s_1. Initialized as empty in case there is no solution at all
S = [];

[n, m] = size(H);

%check if the demensions are right
if abs(n-length(dets))>0
    disp(' Incorrect demensions ! ')
    time = 0;
    return;
end

%Then we do the QR factorization.
[Q, M] = qr(H);

%dets = dets - noise;

%start the MLSD here...
```

```
%Seperate the Q = [Q1, Q2]: Q1 is first m orthonormal columns
%Q2 is last n-m orthonormal columns of Q. And take the m by
%m upper triangular matrix R from the M -- Q2 might be empty.
Q1 = Q(:, 1:m);
Q2 = Q(:, m+1:n);
R = M(1:m, :);

%check if r_(k, k)>0, if not, change the signs of all elements
%in kth row in R and hence the kth column's sign in Q1.
for j = 1:m
    if R(j, j) < 0
        R(j, j:m) = -R(j, j:m);
        Q1(:, j) = -Q(:, j);
    end
end

%Set all the variables initial values.
Y = Q1' * dets; % Y is an m by 1 matrix.
k = m;

%Generate matrices to store intermeditate and final values.
%R_k stores all r'_k(s) and Y stores all y_k(s).
R_k = zeros(m, 1);
```

```
R_k(m, 1) = sqrt(r^2 - norm(Q2' * dets, 2)^2);
Y_k = zeros(m+1, m+1);
Y_k(m, m+1) = Y(m, 1);
UBs_k = NaN(m, 1);

%Using S_k to memorize all values from s_2 to s_m.
S_k = NaN(m, 1);

%code for the algorithm:
while k <= m

    if R(k, k) == 0
        disp(' round off error is too large
        to calculate the solution ')
        break;
    else
        UBs_k(k, 1) = floor((R_k(k, 1)+Y_k(k, k+1))/R(k,k));
        sk = ceil((Y_k(k, k+1) - R_k(k, 1))/R(k, k)) - 1;
    end

    while 1
        sk = sk + 1;
        if sk <= UBs_k(k, 1)
```

```
            S_k(k, 1) = sk;
            if k == 1
                S = [S S_k];
            else
                k = k - 1;
                temp = 0;
                for j = (k+1):m
                    temp = temp + R(k, j) * S_k(j, 1);
                end
                Y_k(k, k+1) = Y(k, 1) - temp;
                R_k(k, 1) = sqrt((R_k(k+1, 1))^2 -
                (Y_k(k+1, k+2) - R(k+1, k+1)*S_k(k+1, 1))^2);
                break;
            end
        else
            k = k +1;
            if k > m
                break; %now, we've got back to the "root".
            else
                sk = S_k(k, 1);
            end
        end
    end
```

```
    %break;
end


time = cputime - time;
```

The LRSD Algorithm Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo April. 11 2008
%
%McMaster University, Department of Computing and Software
%
%Purpose: implement LLL reduction aided sphere decoding
% algorithm
%
%Inputs:
%        dets: received signal vecto with all entries are real
%              the vector must be a column vector
%        H: channel matrix
%
%Output:
%        S: the sub-lattice space containing all the potential
%           solutions time: the time used to run the SD
%           algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [S, time] = lrsd(dets, H)
```

```
%initial the beginning time:
time = cputime;

%S is a matrix memorizeing s_k(s) at each step of k, from s_m
%to s_1. Initialized empty in case there is no solution at all
S = [];

%dets = dets - noise;

[n, m] = size(H);

%check if the demensions are right
if abs(n-length(dets))>0
    disp(' Incorrect demensions ! ')
    time = 0;
    return;
end

%Do LLL reduction to get QR decomposition of the reduced basis
[B, Q, R, U] = LLLReduction(H);

%find the largest vector in the basis
r = norm(B(:, 1), 2);
```

```
for i = 2:m
    temp = norm(B(:, i), 2);
    if r < temp
        r = temp;
    end
end
%r = r/2;

%start the LRSD here...

%Set all the variables initial values.
Y = Q' * dets; % Y is an m by 1 matrix.
k = m;

%Generate matrices to store intermeditate and final values.
%R_k stores all r'_k(s) and Y stores all y_k(s).
R_k = zeros(m, 1);
R_k(m, 1) = sqrt(r^2 - norm(dets, 2)^2 + norm(Q' * dets, 2)^2);
Y_k = zeros(m+1, m+1);
Y_k(m, m+1) = Y(m, 1);
UBs_k = NaN(m, 1);

%Using S_k to memorize all values from s_2 to s_m.
```

```
S_k = NaN(m, 1);

%code for the algorithm:
while k <= m

    if R(k, k) == 0
        disp(' round off error is too
        large to calculate the solution ')
        break;
    else
        UBs_k(k, 1)=floor((R_k(k, 1)+Y_k(k,k+1))/R(k,k));
        sk=ceil((Y_k(k,k+1)-R_k(k,1))/R(k,k)) - 1;
    end

    while 1
        sk = sk + 1;
        if sk <= UBs_k(k, 1)
            S_k(k, 1) = sk;
            if k == 1
                S = [S S_k];
            else
                k = k - 1;
                temp = 0;
```

```
                for j = (k+1):m
                    temp = temp + R(k, j) * S_k(j, 1);
                end
                Y_k(k, k+1) = Y(k, 1) - temp;
                R_k(k, 1) = sqrt((R_k(k+1, 1))^2 -
                (Y_k(k+1,k+2)-R(k+1,k+1)*S_k(k+1,1))^2);
                break;
            end
        else
            k = k +1;
            if k > m
                break; %now, we've got back to the "root".
            else
                sk = S_k(k, 1);
            end
        end
    end
    %break;
end

%To find the solution via U^-1 * z as in the algorithm

time = cputime - time;
```

```
if isequal(S, [])
    return;
else
    S = inv(U)*S;
end
```

The Optimal Solution Standard Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo, April 11, 2008
%
%McMaster University, Department of Computing and Software
%
%Purpose: Choose an optimal solution from a given sublattice
%
%Inputs:
%         S: the sublattice points to choose from
%         H: channel matrix
%         dets: the detected signal
%
%Output:
%         signal: single vector represents optimal solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%this function is to check the "optimal" solutions inside a
%sublattice. it choose a lattice point such that Hs-v is
%minimized and is from the L-PAM
```

```
%here in particular is 4-PAM or 8-PAM.
function  signal = optimalsoln(S, H, dets)

[m, n] = size(S);
[mh, nh] = size(H);

if abs(m-nh) > 0 | n == 0
    signal = zeros(m, 1);
    return;
end

%choose signals from L-PAM constellation contained in this
%sublattice

sublattice = [];

switch m
    case 4
        for i=1:n
            %check each column's entries
            signal = S(:, i);
            %set a flag to indicate if need to add this signal
            flag = 1;
```

```
        for j=1:m
            temp = signal(j, 1);
            temp1 = abs(temp);

            if temp1 == 1 | temp1 == 3
            %test if the entries from 4-PAM
                signal(j, 1) = temp;
            else
                flag = 0;
                break;
            end
        end

        %if we have come to the last
        if flag == 1
            sublattice = [sublattice signal];
        end
    end
case 8
    for i=1:n
        %check each column's entries
        signal = S(:, i);
        %set a flag to indicate if need to add this signal
```

```
            flag = 1;
            for j=1:m
                temp = signal(j, 1);
                temp1 = abs(temp);

                %test if the entries from 4-PAM or 8-PAM
                if temp1 == 1|temp1 == 3|temp1 == 5|temp1==7
                    signal(j, 1) = temp;
                else
                    flag = 0;
                    break;
                end
            end

            %if we have come to the last
            if flag == 1
                sublattice = [sublattice signal];
            end
        end
    otherwise
        disp(' This is not a 4-PAM or 8-PAM !!! ')
        return;
end
```

```
%choose optimal solution by the ML
[ms, ns] = size(sublattice);

%if no solution in the sublattice, then return a 0 vector
if ns == 0
    signal = zeros(m, 1);
    return;
end

%otherwise we find the optimal one
if ns == 1
    signal = sublattice;
    return;
else
    signal = sublattice(:, 1);
    current = norm((H*sublattice(:, 1) - dets), 2);
end

for i=2 : ns
    hypersig = sublattice(:, i);
    temp = norm((H*hypersig - dets), 2);
    if temp < current
```

```
        signal = hypersig;
    end
end
```

Random Signal Generator Implementation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo, April 11, 2008
%
%McMaster University, Dept. of computing and software
%
%Purpose: generate a random signal from either 4-PAM
%         or 8-PAM constellation, i.e. {-+1, -+3, -+5, -+7}
%
%Input:
%       m: m=4 or m=8, otherwise an error message displayed
%
%Output:
%       signal: a column vector with length either 4 or 8
%       and entries from the correspondent constellation
%
%Precondition: m=4 or m=8
%Postcondition: a column vector signal with lenthg 4 or 8
%
%Note: the precodition is not checked in this program, it
%      is considered the tester has made sure precodition
%      is met
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function signal = signalGenerator(m)

%if it is a 4-PAM constellation, i.e. m=4:
signal = mod(round(m*rand(m, 1)), m);

for i=1 : m
    switch signal(i, 1)
        case 0
            signal(i, 1) = 3;
        case 1
            signal(i, 1) = 1;
        case 2
            signal(i, 1) = -1;
        case 3
            signal(i, 1) = -3;
        case 4
            signal(i, 1) = 5;
        case 5
            signal(i, 1) = -5;
        case 6
```

```
            signal(i, 1) = -7;
        case 7
            signal(i, 1) = 7;
        otherwise
            disp('something is wrong')
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The Main Test Program

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin Cai Guo, April 11, 2008
%
%McMaster University, Department of Computing and Software
%
%Purpose: compare the Maximum Likelihood Sphere Decoding
%         performance with that of LLL reduction aided
%         sphere decoding's
%
%Inputs:
%         m: transmitted signal length, m=4 or m=8 when use
%            4-PAM or 8-PAM respectively
%         L: channel order
%         snr: signal to noise ratio
%         pr: probability needed to find the optimal solution
%
%Outputs:
%         serML: symbol error rate using Maximum Likelihood
%                Sphere Decoding algorithm
%         serLR: symbol error rate using LLL reduction aided
%                Sphere Decoding algorithm
```

```
%          timeML: time used by Maximum Likelihood Sphere
%                  Decoding algorithm to find optimal solution
%          timeLR: time used by LLL reduction aided Sphere
%                  Decoding algorithm to find optimal solution
%
%Precodition: m=4 or m=8 for 4-PAM or 8-PAM respectively,
%             iniRadius is obtained throught the probability
%             table with degree of freedom set as 3 and
%             probability can only be (0.95 - 0.99)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function[serML,serLR,timeML,timeLR]=finaltest(m, L, snr, pr)

%generate the channel matrix
h = randn(L, 1);

n = L + m - 1;

H = zeros(n, m);

for i = 1:m
    H(i: i+L-1, i) = h;
end
```

```
%find scalar to satisfy the power constrain function
%constellation, i.e. 4-PAM or 8-PAm
switch m
    case 4
        scal = 1/sqrt(5);
    case 8
        scal = 1/sqrt(21);
    otherwise
        ser1 = [];
        timeML = [];
        disp(' Undefined PAM constellation ')
        return;
end

%calculate the initial searching radius
if pr >= 0.99
    iniRadius = 3.7817;
else if pr >= 0.975
        iniRadius = 3.116;
    else
        iniRadius = 2.605;
    end
```

```
end
r = sqrt(iniRadius);

%run both algorithms 10,000 times to find the SER and
%average time for different randomly generated signal
%from 4-PAM or 8-PAM and Gaussian noise

countML = 0;
countLR = 0;
timeML = 0;
timeLR = 0;

for i=1 : 1000

    clear noise signal dets s1 s2 SML SLR;

    switch m
        case 4
            signal = signalGenerator(m);
        case 8
            signal = signalGenerator(m);
        otherwise
            serML = 0;
```

```
        disp(' Undefined PAM constellation ')
        return;
end
noise = randn(n, 1)/((10^(snr/10))*scal);
%noise = zeros(n, 1);

%calculate the hypothetical received signal
dets = H*signal + noise;

%use the MLSD to solve the transmitted symbols
[SML, time1] =  mlsd(dets, H, r);
s1 = optimalsoln(SML, H, dets);

timeML = timeML + time1;
if isequal(s1, signal)
    countML = countML + 1;
end

%use LLL reduction aided SD solve transmitted symbols
[SLR, time2] = lrsd(dets, H);
s2 = optimalsoln(SLR, H, dets);

timeLR = timeLR + time2;
```

```
    if isequal(s2, signal)
        countLR = countLR + 1;
    end
end

serML = 1 - countML/1000;
serLR = 1 - countLR/1000;

%average time spend
timeML = timeML/1000;
timeLR = timeLR/1000;
```

The Graph Plot Program

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Author: Jin cai Guo  April 11, 2008
%
%Purpose: plot the SER and efficiency comparison graph
%
%Input:
%       sr: the single row vector contains the SNR values
%       pr: probability you want to test on
%
%Output:
%       SER performance graph of MLSD and LRSD
%       Efficiency performance graph of MLSD and LRSD
%
%Note: the efficiency graph in terms of time spend in ms
%      has been commented off, but can be turned back.
%
%precodition: the pr can only be (0.95 - 0.99)
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function graphplot(sr, pr)
```

```
n = length(sr);

%y-axis
%Y1 = zeros(1, n);
%Y2 = zeros(1, n);
Z1 = zeros(1, n);
Z2 = zeros(1, n);

for i=1:n
    clear serML serLR timeML timeLR;
    [serML,serLR,timeML,timeLR]=finaltest(4,3,sr(i),pr);
    %Y1(1, i) = serML;
    Z1(1, i) = timeML;
    %Y2(1, i) = serLR;
    Z2(1, i) = timeLR;
end


%plot(sr,Y1,'--rs','LineWidth',1)
%hold all
%plot(sr,Y2, '-gs','LineWidth',2)
%xlabel('SNR')
```

```
%ylabel('SER')
%title('SER performance of MLSD and LRSD')


plot(sr,Z1,'--rs','LineWidth',1)
hold all
plot(sr,Z2, '-gs','LineWidth',2)
xlabel('SNR')
ylabel('TIME (/ms)')
title('Efficiency (TIME) performance of MLSD and LRSD')
```